ENTWICKLUNG EINER LOKALEN SUCHMASCHINE MIT DEM WEB APPLICATION SERVER VON ORACLE 8

von

Jury Guido Weide

Othestraße 32a 51702 Bergneustadt

Matr.-Nr.:10227216

Fachbereich: Informatik
Fachhochschule Köln, Abt Gummersbach
Diplomarbeit zur Erlangung des akademischen Grades

Diplom-Informatiker

1. Prüfer: Prof. Dr. Faeskorn-Woyke

2. Prüfer: Prof. Dr. Westenberger

Gummersbach, im September 1999

INHALTSVERZEICHNIS

0. Einführung

Die vorliegende Diplomarbeit beschäftigt sich mit der neusten Version des Web Application Servers in der Version 3.0 von Oracle. Der Web Application Server ist ein Programmpaket für die Entwicklung und Implementation von serverseitigen Anwendungsprogrammen, die über das Internet benutzt werden können. Das entsprechende Anwendungsprogramm wird nur vom Webserver ausgeführt. Für die Ausführung einer mit Hilfe des Web Application Server erstellten Anwendung wird auf der Clientseite nur ein Internetbrowser benötigt. Der Web Application Server wurde speziell daraufhin ausgelegt, daß die mit seiner Hilfe entwickelten Anwendungen ihre Daten aus einer Datenbank beziehen können, schließlich ist das Hauptprodukt von Oracle einer der weltweit am meisten eingesetzten Datenbanksysteme. Für den Anwendungsentwickler ermöglicht der Web Application Server also Anwendungen für das Internet zu erstellen, wobei der Entwickler auf eine breite Palette von Programmiersprachen, die mit den Web Application Server - Komponenten kompatibel sind, zurückgreifen kann. Das sind z.B. Perl, C++, Java und natürlich die für Oracledatenbanken benutzte Sprache PL/SQL.

Da der Web Application Server die Internettechnologie, sowie die Datenbanktechnologie berührt, werden in den ersten Abschnitten nicht nur der Nutzen dieser Plattform im Rahmen der Client/Server Architektur gezeigt, sondern auch Grundlagen des Internets und relationaler Datenbanken vorgestellt. Dieses ist notwendig, um die Abläufe von Web Application Server basierten Anwendungen zu verstehen. In den weiteren Kapiteln werden die einzelnen Bauteile des Web Application Server - Programmpaketes vorgestellt und ihre Funktionen erklärt. Hierbei wird kurz auf die Installation und Konfiguration der Web Application Server Plattform eingegangen. Anschließend soll das Zusammenspiel der Web Application Server - Elemente mit Hilfe einer Beispielanwendung gezeigt werden.

1. Der Web Application Server und seine Schnittstellen

1.1. Einsatzgebiete des WAS

Seitdem immer mehr PCs in den Unternehmen eingesetzt werden, hat sich das traditionelle „ Terminal – Host „ Schema in der EDV gründlich verändert. Früher war ein EDV Arbeitsplatz mit einem Terminal ausgerüstet, eine Bildschirmstation, die nichts anderes konnte als Text darzustellen. Diese Stationen waren an einem Großrechner angeschlossen auf dem jegliche Verarbeitung stattfand. Mit dem Einzug des PCs bekamen die Benutzer ein Werkzeug an die Hand, welches selbst umfangreiche Programme und Anwendungen verarbeiten kann. Da es wünschenswert ist, Daten auszutauschen, werden diese PCs miteinander vernetzt. Der früher übliche Großrechner wurde durch Middleframerechner ersetzt – die Client/Server Architektur entstand. Diese Server übernehmen die Speicherung von zentralen Daten, z.B. in Datenbanken und führen entsprechende Anwendungen aus, um diese als Dienst an den Clients weiterzugeben. Die heutigen Client PCs nutzen die Dienste des Servers, können aber selbst umfangreiche Anwendungen und Daten speichern und ausführen. Anwendungen, die früher nur auf dem Großrechner liefen, werden so dezentralisiert d.h. verteilt. Jeder Benutzer eines Clients hat die Möglichkeit individuelle Anwendungen auf seinem PC einzurichten und ihn so auf seine Bedürfnisse zu konfigurieren. Dies geht inzwischen so weit, daß der Begriff der individuellen Datenverarbeitung (IDV) entstand. Mit der Zeit erkannte man jedoch, daß diese Dezentralisierung auch mit Nachteilen behaftet ist. Hierbei ist der Hauptgrund in den enormen Kosten für die Hardware, Softwarepflege und Benutzerunterstützung zu sehen. Selbst in einem mitelständigen Unternehmen werden nicht selten mehr als 1000 Client PCs benutzt.

Wenn man sich nun vorstellt, daß jedes Softwarepaket auf jeden PC eingerichtet und aufgerüstet werden muß, die Hardwareausstattung jedes PCs von Zeit zu Zeit aktualisiert werden muß, Software-, Hardware- und Benutzerfehler behoben werden müssen, sowie Virenschutz und Datensicherheit gewährleistet sein soll, ist es einzusehen, daß diese Aufgaben jede Menge Personal- und damit Kostenaufwand erfordern. Um diesen Aufwand zu vermeiden, kann man die Anzahl von Anwendungen auf den Client PCs verringern, in der Literatur auch als „ thin client “ Konzept bezeichnet. Im äußersten Fall sollte hier der Client PC nur Anwendungen zur Aufbereitung von Informationen ausführen, also eine Benutzeroberfläche, um die Daten für den Benutzer darzustellen. Solche Benutzeroberflächen sind heute schon auf vielen PCs installiert, die Internetbrowser. Das Internet bietet hier noch mehr Vorteile: Die Benutzeroberfläche, also der Browser, ist weitgehend standardisiert und verbreitet, das Internet benutzt zum Datenaustausch ein Industriestandardprotokoll, das TCP/IP, welches heute in vielen Firmennetzen eingesetzt wird und schließlich ist das Internet ein weltweit verfügbares Netz. Genau hier setzt der Web Application Server von Oracle auf. Der Benutzer erhält die Möglichkeit, Anwendungen auf einem Webserver auszuführen und benötigt hierfür nur die Browsersoftware auf seinem PC. Anwendungen werden nur auf dem Server ausgeführt, der seinerseits durch einen Listener mit dem Internet verbunden ist. Die Bestandteile des WAS sind in der Lage Programme auszuführen mit deren Hilfe Internetseiten generiert werden, die wiederum der Browser für den Benutzer aufbereiten kann. Hierbei unterstützt die WAS Plattform mehrere Programmiersprachen, wie C, Perl und Java.

Das Besondere am WAS ist jedoch die Möglichkeit, neben den oben genannten Programmiersprachen, über die Oracle Datenbanksprache PL/SQL datenbankbasierte Anwendungen zu erstellen. Das WAS ist hierbei ein offenes und plattformunabhängiges System. Der Web Application Server kann auch zusammen mit Fremddatenbanken, also nicht Oracle Datenbanken verwendet werden.

In der Arbeit soll in erster Linie die Verwendung des WAS mit der Oracle 8 Datenbank gezeigt werden. Hierzu ist es nötig sich mit der Datenbanksprache PL/SQL auseinanderzusetzen. Dieses ist im Kapitel 1.3. „ relationale Datenbanken „ ,soweit es für das Verständnis des WAS wichtig ist, gemacht worden. Da sich mit dem WAS Internetanwendungen erstellen lassen, ist es zunächst aber erforderlich, näher auf die Architektur und Funktion des Internets und seines HTTP Protokolls einzugehen.

1.2. Das Internet und sein Aufbau

Der Grundstein des Internet wurde Mitte der 60'er Jahre in den USA gelegt. Das Ziel war es ein Rechnernetz aufzubauen, daß keine direkte Verbindung zwischen dem Sender und Empfänger benötigt, um Daten auszutauschen. 1969 wurde das "APRANET" für das amerikanische Verteidigungsministerium verwirklicht. Dieses Netz wurde im Verlaufe der nächsten acht Jahre von vier auf 111 Knoten erweitert. Zusätzlich wurden öffentliche Netzwerke nach diesem Vorbild aufgebaut. Die meisten dieser Netze schlossen sich Ende der 80'er Jahre zusammen, woraus dann das Internet entstand. Bis heute hat die Anzahl, der im Internet zusammengeschlossenen Rechner explosionsartig zugenommen. Der Datentransport im Internet bildete und bildet bis heute in der Transportschicht das Transmission Control Protocol (TCP). Damit die Rechner innerhalb des Netzes kommunizieren können, benötigt jedes Gerät eine eindeutige Adresse. Diese wird für das TCP Protokoll IP Adresse genannt und die Datenaustauschmethode folglich TCP/IP. Die IP Adresse besteht aus einer 4 Byte langen Ziffer, wobei die Bytes jeweils voneinander durch einen Punkt getrennt werden. Die IP Adresse setzt sich aus zwei Hauptteilen zusammen, die Netzwerknummer, die zentral vergeben wird, um die Eindeutigkeit jedes Anschlusses zu garantieren und der Hostnummer, welche die Bezeichnung ein einzelnes Geräts innerhalb eines Hostnetzwerks darstellt. Die Daten, die durch das TCP Protokoll versendet werden, werden durch das Protokoll in Datenpakete unterteilt. Diese Datenpakete, die eine Länge von bis zu 1500 Zeichen haben können, werden durchnumeriert und erhalten eine Absender IP Adresse und eine Empfänger IP Adresse.

Die Datenpakete können so über verschiedene Wege den Empfänger erreichen, Sender und Empfänger sind also nicht direkt miteinander verbunden.. Vielmehr nehmen die Datenpakete unterschiedliche Wege d.h. über verschiedene Netzknotenrechner (Router). Beim Empfänger werden sie mit Hilfe der Numerierung wieder in der richtigen Reihenfolge zusammengesetzt. Auf ihrem Weg vom Sender zum Empfänger können einzelne Datenpakete verloren gehen oder nicht lesbar sein. Dieses wird vom Empfänger festgestellt, so daß er die fehlenden Pakete vom Sender nachfordern kann. Nach dem erfolgreichen Transport der Daten wird diese „indirekte„ Verbindung zwischen Sender und Empfänger abgebrochen. Der wohl wichtigste Grund für die starke Verbreitung von Internetanschlüssen ist ein Dienst mit dem Namen world wide web (WWW). Mit diesem Dienst ist es möglich nicht nur reine Texte zu übertragen, sondern auch Bilder, Grafiken, Töne, Animationen und andere multimediale Daten, sowie sogenannte Links. Das zu Grunde liegende Verfahren, um diese Daten im WWW austauschen zu können, wird HTTP (Hypertext Transfer Protocol) genannt. Hierbei handelt es sich um ein "Application Layer Protocol", welches über das TCP/IP läuft.

1.2.1 Das HTTP Protokoll

Ein Datenaustausch im WWW erfolgt normalerweise in Form eines Frage – Antwort Ablaufs, so ist das HTTP Protokoll auch ein Frage – Antwortprotokoll. Der Benutzer gibt in seinem Browser eine Adresse an, wo die gewünschten Informationen bzw. die gewünschte Webseite im Netz liegt.

Die Informationen liegen normalerweise auf einem Webserver und sind dort in irgend einem Verzeichnis abgelegt. Solch eine Adresse wird im WWW als uniform resource location bzw. URL bezeichnet. Ein Teil der URL ist nichts anderes, als ein Aliasname für die IP-Adresse des Servers, auf dem die gewünschten Daten liegen. Die Verwaltung der Anfragen nach WWW-Resourcen übernimmt auf der Serverseite der Weblistener. Der Ablauf einer solchen Internetanfrage sieht normalerweise so aus: Der Benutzer gibt die URL entweder manuell in den Browser ein, oder ein Link verweist auf die URL. Dann wird eine Verbindung zu dem in der URL bezeichneten Webserver aufgebaut. Die Listenersoftware auf dem Webserver nimmt alle an den Server gerichteten Anfragen entgegen und übersetzt die URL in eine Pfadangabe, in der sich die angefragte Resource befindet. Ist diese vorhanden und der Anfrager berechtigt auf sie zuzugreifen, werden die Daten zum Benutzerbrowser gesendet. Nachdem die Übertragung der Daten abgeschlossen ist wird die Verbindung zwischen dem Server und dem Browser abgebrochen.

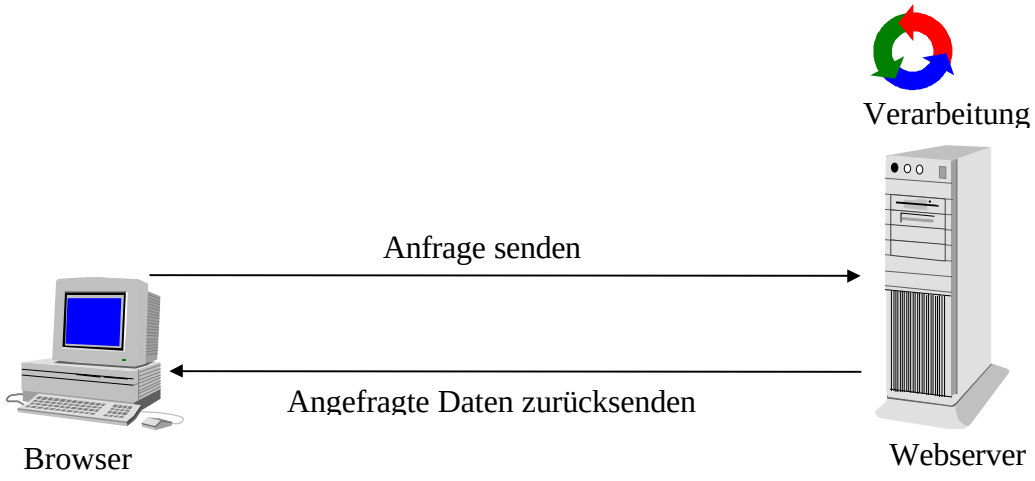


Bild 1 : Ablauf einer WWW Verbindung

Die Daten, die zwischen Client und Serverrechner verschickt werden, sind in einer bestimmten Form aufgeteilt. Anfragen sowie Antworten sind hier in einen Kopfteil und einen Hauptteil getrennt. Der wichtigste Teil der Anfrage ist hierbei der Anfragekopf, wobei der Anfragehauptteil optional ist. Der Anfragekopf muß aus einem Kommando bestehen. Dieses ist die Spezifikation der Methode, die der Server auf die vom Browser verlangte Resource anwenden soll. In der HTTP 1.0 Originalversion gab es nur die beiden Kommandos „post“ und „head“. Bei dem head Kommando sollte der Server nur einen Antwortkopf senden, während bei dem post Kommando eine vollständige Antwort vom Browser verlangt wird. Eine weitere wichtige Methode ist die “ get “ Methode, die erst bei neueren Versionen des HTML eingesetzt wird. Sie gibt ähnlich wie die „post “ Methode den Kopf und den Inhalt der Antwort wieder, benutzt aber die Anfrage dazu eine Antwort zu generieren. Die „get“ Methode wird daher bei dynamischen Webseiten benutzt, wie z.B. bei der Erstellung von Webanwendungen mit dem Web Application Server. Zusätzlich benötigt der Listener den Standort der nachgefragten Resource. Diese wird als uniform resource identifier oder URI bezeichnet. Die URI ist nichts anderes als die URL ohne die Host und Portinformationen, da der Server für die Anfrage ja schon angesprochen wurde. Aus dieser URI kann der Server nun den Pfad und die dort gespeicherte angefragte Datei heraussuchen. Die Antwort des Servers besteht ebenfalls aus einem Kopf, der u.a. Informationen über die nachfolgenden Daten enthält und den Browser darüber informiert, wie er mit den Daten umzugehen hat. Die eigentlichen Daten werden dann im Hauptteil gesendet. Die Daten werden hier auch in einem gewissen Format abgefaßt, nämlich in der Visualisierungssprache des WWW, die Hypertext Markup Language bzw. HTML. Mit ihrer Hilfe kann der Browser Texte und andere Daten interpretiert und für den Benutzer aufbereitet.

1.2.2. Textformatierung mit HTML

Das HTML Format ist ein reines Textformat. HTML Dateien können mit einem normalen Texteditor geschrieben und mit dem Anhängsel .html oder .htm abgespeichert werden. Die Formatierung des Textes erfolgt mit sog. Tags, welches festgelegte Befehle sind, die in eckigen Klammern gesetzt werden. Der Text wird in diese Tags eingebettet. Viele Tags treten in Paaren auf – ein Anfangstag, gefolgt vom eigentlichen Text und ein Endtag. Das Tag <B> formatiert z.B. einen Text in Fettbuchstaben bis das Tag </B> diese Formatierung wieder aufhebt. Eine HTML Zeile könnte demnach so aussehen: <B>Dieser Text ist Fett</B>. Es gibt viele Textformatierungstags z.B. für Überschriften, Kursivschrift, Unterstreichungen usw. Tags können aber auch andere Funktionen erfüllen. Mit ihrer Hilfe können Bilder eingebunden werden, Verweise auf andere URL (sog. Links) gesetzt werden oder Benutzereingaben verwaltet werden. Um z.B. ein Bild in eine HTML Seite einzubinden, muß es erst einmal auf dem Server unter irgend einem Pfad gespeichert sein. Dies könnte z.B. ein Bild im gif Format sein, welches unter dem virtuellen Pfad /images/bild.gif vom Weblistener verwaltet wird. Der physikalische Pfad, unter dem das Bild gespeichert ist wird vom Weblistener unter einem virtuellen Pfad verwaltet. Das Tag um ein Bild in eine Webseite einzubinden ist das <IMG> Tag. Dieses Tag enthält ein Attribut welches auf dem Pfad des einzubindenden Bild verweist. Eine Zeile für das Einbinden des Bildes könnte nun sein: <IMG SRC="images/bild.gif">. Ein weiterer wichtiger tag ist der "Anchor Tag". Mit ihm ist es möglich einen sog. Link auf eine andere URL zu erstellen.

Ein Link ist ein Text- oder Bildobjekt,. welches einen Verweis auf eine andere URL oder eine andere Textstelle auf der aktuellen Webseite enthält. Das Format des Anchor Tags ist <A>...</A>. Auch das Anchor Tag beinhaltet ein Attribut, das HREF-Attribut für Hyperreference. Mit Hilfe dieses Attributs spezifiziert man die URL, die angesprochen werden soll, wenn der Benutzer auf die Text- oder Bildstelle klickt. Mit den bisher vorgestellten html Befehlen wird nichts als die URL an den Webserver geschickt – der Benutzer hat bisher noch nicht die Möglichkeiten eigene Eingaben zu machen und sie an ein Programm auf dem Webserver zu senden. Damit dies möglich wird, unterstützt html sogenannte Eingabeforms. Diese sind Formatierungen, die der Entwickler der Webseite benutzen kann, um Eingabemasken für den Benutzer zu erstellen. Außerdem können sie die Benutzereingaben speichern und nach dem Absenden der vom Benutzer ausgefüllten Eingabemaske an den Server übergeben. Diese Eingabemasken oder Forms können Listen, Menüs und Texteingabefenster enthalten. Solche Eingabemasken werden ebenfalls mit Hilfe von speziellen HTML-Tags erstellt. Eine Eingabemaske wird zwischen den Tags <FORM>...</FORM> programmiert. Die Masken besitzen zwei Hauptattribute, nämlich METHOD und ACTION. Als Methode kann entweder "get" oder "post" angegeben werden – sie spezifiziert die Methode, mit welcher der Server Anwendungsprogramm, in dem die Eingabedaten benötigt werden, aufgerufen werden soll (siehe auch S.8 "post" und "head"). Das ACTION Attribut spezifiziert entweder ein CGI-Script oder ein Web Application Server cartridge. Ein Cartridge ist ein Programm, daß eine bestimmte Programmiersprache unterstützt.

Das Web Application Server Paket enthält vorgefertigte Cartridges, welche PL/SQL, Perl und ODBC für nicht Oracle Datenbanken unterstützen. Cartridges sind Teil der Web Application Server Architektur und werden im Kapitel 2 - Aufbau und Funktion des Web Application Servers - näher erklärt. Zwischen den <FORM>...</FORM> Tags können Texte, andere Formattags oder Bilder eingefügt werden um die Eingabeseite zu gestalten. Die eigentlichen Eingabetypen sind jedoch Texteingabefelder, Checkboxen, Radioboxen, Drop-Down-Menüs und Multielementlisten. Diese werden mit Hilfe des <INPUT> Tags eingebunden, welches verschiedene Attribute haben kann, wie z.B. der Typ des Eingabeelements. Das NAME Attribut muß bei allen INPUT Eingabetypen angegeben werden, außer bei dem Submiteingabetyp, hier ist er optional, da dieser Eingabetyp das Versenden der Eingabeseite auslöst, also nicht unbedingt eigene Daten enthalten muß. Das NAME Attribut kann man als eine Art Variable auffassen, in der sich die Eingabe des Benutzers ablegen läßt. Löst der Benutzer durch einen Submitbefehl das Versenden der Eingabedaten aus, so werden seine Eingaben als name/value Paar an den Webserver versendet. Name spezifiziert hierbei das Element, welches im Input Attribut NAME festgelegt wurde und value ist der Wert, den dieses Element durch die Benutzereingabe erhalten hat, also die Benutzereingabe selbst. Der Benutzer kann auf solch einer Eingabemaske seine Daten eingeben und muß danach das eigentliche Senden der Daten durch einen Submitbefehl auslösen. Dieses kann vom Entwickler der Eingabemaske durch einen Submitknopf oder einer –graphik erreicht werden, auf den der Benutzer klicken kann.

Es würde diese Arbeit sprengen, eine komplette Übersicht aller HTML-Befehle zu geben, da dieses Thema ganze Bücher füllt, auf die an dieser Stelle für eine komplette HTML-Übersicht verwiesen werden soll. Die Erklärung dieser grundlegenden HTML-Befehle ist jedoch notwendig gewesen, da der Web Application Server bei der Verarbeitung eines PL/SQL Programms die Ausgabe im html Format generieren kann. Der Web Application Server unterstützt hierzu Libraries oder Pakete wie htp oder htf Funktionen, die später erklärt werden. Zunächst jedoch eine Einführung in relationale Datenbanken und der 4GL-Sprache SQL bzw. PL/SQL mit welcher die Beispielanwendung ,ab Kapitel 4 dieser Arbeit, hauptsächlich erstellt wurden.

1.3. Relationale Datenbanken

1.3.1. Aufbau relationaler Datenbanken

Durch die elektronische Datenverarbeitung ist es nicht nur möglich Algorithmen für die Abarbeitung von Daten oder die Steuerung von Prozessen zu benutzen, sondern durch die Fülle der Informationen in der heutigen Zeit ist eine Verwaltung dieser Daten erforderlich. Um Informationen möglichst effektiv zu speichern und auf sie zugreifen zu können, entstanden in den 60er Jahren die ersten Datenbanken. Mit ihnen sollte es möglich sein Daten unabhängig von bestimmten Programmen, die zur damaligen Zeit noch nicht in dem Maße wie heute standardisiert waren, zu speichern, so daß auf sie durch eine einheitliche Schnittstelle zugegriffen werden kann. Außerdem sollten mehrere Benutzer parallel auf die Daten zugreifen können und die Zugriffsmöglichkeiten sollten aus Datensicherheitsgründen für jeden Benutzer flexibel festzulegen sein. Eine Datenbank besteht aus der Datenbasis, welche die eigentlichen Informationen aufnimmt und dem Datenbankverwaltungssystem, um auf die Datenbasis zugreifen zu können, sie zu pflegen und Regeln der Datenbankbenutzung durchzusetzen. Es entstanden mit der Zeit verschiedene Konzepte und Ansätze für Datenbanken, die wichtigsten Modelle sind das hierarchische Datenbankmodell, in dem Daten in Baumstrukturen abgelegt werden, dem Netzwerkmodell, daß ähnlich wie das hierarchische aufgebaut ist, wo aber einzelne Blätter eines Baumes mit einem Blatt eines anderen Baumes verbunden sein können, dies wird bei beiden Modellen durch Adressenverweise realisiert und schließlich der relationale Datenbankansatz, zu welchem auch die in dieser Arbeit verwendete Oracle-Datenbank gehört. Die relationalen Datenbanken haben hierbei einige Vorteile gegenüber den anderen Konzepten.

Mit ihnen ist wird Datenredundanz minimiert, d.h. die Daten und Datenbeziehungen können zentral und einmalig abgelegt werden, welches Ressourcen, wie Speicherplatz spart, die Geschwindigkeit des Datenzugriffs erhöht und die Übersichtlichkeit der Daten fördert. Relationale Datenbanken sind außerdem flexibler zu handhaben, eine Änderung eines Datensatzes zieht bei ihnen nicht die Neuprogrammierung des gesamten Datenbestandes nach sich, wie dies durchaus bei den anderen Ansätzen vorkommen kann. Ab 1980 kamen die ersten relationalen Datenbanken auf den Markt, wie schließlich auch die erste Oracle-Datenbank. Bei relationalen Datenbanken werden die Daten in Tabellen abgebildet. Eine Tabelle kann hierbei als eine Relation aufgefaßt werden, die Attribute der Relation entsprechen hierbei den Tabellenspalten , ein Tupel der Relation entsprechen einer Tabellenzeile. Da es sich um einen relationalen Ansatz handelt ist die Reihenfolge der Zeilen bzw. der Spalten der Tabelle nicht definiert. Die mathematischen Eigenschaften von Relationen sind auch in den Tabellen relationaler Datenbanken wiederzufinden. Eine Relation darf keine zwei oder mehr identische Tupel aufweisen, was für die Tabellen relationaler Datenbanken bedeutet, daß keine zwei Zeilen einer Tabelle komplett identisch sein dürfen. Ein Attribut einer Relation muß atomar sein, d.h. es darf nur eine Ausprägung annehmen, was wiederum für die Tabellen bedeutet, daß keine zusammengesetzten Datentypen in den Spalten stehen dürfen. Einzelne Attribute können einen nicht definierten Wert besitzen, was bei relationalen Datenbanken als NULL Wert bezeichnet wird. Jede Zeile einer Tabelle bzw. Tupel einer Relation verfügt über ein oder mehrere Attribute, die diese Zeile eindeutig identifiziert. Das oder die Attribute, die eine Zeile eindeutig identifizieren wird als Primärschlüssel bezeichnet.

Da der Primärschlüssel diese Identifikationsaufgabe hat, darf er natürlich nicht, auch nicht ein Teil von ihm, nicht definiert sein, also keinen NULL Wert annehmen. Erfüllen die Tabellen und der Primärschlüssel in den Tabellen diese Anforderung, so ist die Entity Integrität der Datenbank gewährleistet. Damit die Tabellen untereinander in Bezug gesetzt werden können tauchen Primärschlüssel einzelner Tabellen in anderen Tabellen auf, in denen sie als Fremdschlüssel bezeichnet werden. Ist eine Datenbank aus mehrere Tabellen oder Relationen zusammengesetzt, sollte jede Tabelle mindestens einen Fremdschlüssel besitzen, der eine andere Tabelle der Datenbank referenziert, die Datenbank besitzt dann auch die referenzielle Integrität. Da per Definitionem ein Fremdschlüssel immer ein Primärschlüssel einer anderen Tabelle sein muß, ist es beim Aufbau einer Datenbank zunächst schwer einzusehen, wie ein Fremdschlüssel ohne Primärschlüssel auftauchen kann. Wird aber nach der Erstellung der Datenbank ein Primärschlüssel gelöscht und ist dieser Primärschlüssel gleichzeitig Fremdschlüssel in einer anderen referenzierten Tabelle, so kann nach der Löschung nicht mehr auf sie Bezug genommen werden, d.h. es besteht keine referentielle Intigrität mehr. Aber nicht nur die Integrität einer relationalen Datenbank spielt eine wichtige Rolle, auch die Effizienz, d.h. die Geschwindigkeit des Datenzugriffs und der Speicherplatz, welche die Daten beanspruchen sind für den optimalen Datenbankentwurf wichtig. Um redundante, d.h. überflüssige Daten und Datenbeziehungen und inkonsistente Daten vorzubeugen und eine leichtere Pflege der Daten zu gewährleisten gibt es bei der relationalen Datenbanktheorie den Begriff der Normalisierung. Hierbei geht es um die Untersuchung von Relationen (Tabellen), Attribute (Spalten) und deren Abhängigkeit untereinander.

In der Regel sind beim praktischen Datenbankentwurf die ersten drei Normalformen zu beachten, während in der Theorie auch die 4. und 5. Normalform existieren, diese sollen aber hier nicht weiter beschrieben werden. Eine Tabelle liegt in der ersten Normalform, auch als 1NF bezeichnet, vor wenn die Attribute der Tabelle aus einer einzigen Information besteht. Dies ist leicht einzusehen, da ja auch in einer Relation die Attribute, wie vorhin beschrieben, atomar sein müssen. Bei der 2NF ist Voraussetzung das zum einen die Tabelle in 1NF vorliegt und alle Spaltenwerte hängen nur vom Primärschlüssel ab, d.h. eine Tabelle sollte keine Fremdinformationen enthalten, die ja unabhängig vom Primärschlüssel sind. In der dritten Normalform ist die Voraussetzung, daß sie die 1NF und die 2NF erfüllt und zusätzlich müssen die Spalten vollständig vom Primärschlüssel abhängen. Die Gefahr die 3NF nicht zu erreichen ist natürlich nur gegeben, wenn die Tabellen einen aus mehren Attributen zusammengesetzten Primärschlüssel besitzt. Um auf relationale Datenbanken zuzugreifen, zu implementieren, zu pflegen und die Daten abzurufen gibt es bei Oracle und auch anderen relationalen Datenbanken die strukturierte Abfragesprachen oder Structured Query Language bzw. SQL.

1.3.2. Grundlegendes zu SQL

Als Standardsprache für relationale Datenbanken hat sich die nicht prozedurale Sprache SQL als Standart durchgesetzt. Mit SQL teilt man einer relationalen Datenbank mit, welche Daten abgerufen oder geändert werden sollen, anders als bei prozeduralen Sprachen, mit denen man dem System mitteilt, wie Daten manipuliert werden sollen. Das original SQL besitzt keinerlei Programmierungskonstrukte, wie Schleifen oder Verzweigungsanweisungen. Wie später noch gezeigt werden soll, besitzt das PL/SQL von Oracle solche Konstrukte sehr wohl, aber sie sind eine Oracle spezifische Erweiterung des SQL. Das original SQL kennt auch keine benutzerdefinierten Datentypen, wie das bei objektorientierten Programmiersprachen der Fall ist – auch hier bietet das Oracle PL/SQL ab der Version 8 Erweiterungen, wie etwa der Record Datentyp. Das SQL unterteilt sich in drei Hauptkomponenten, der Data Definition Language kurz DDL für die Definition der Datenstruktur, die Data Control Language kurz DCL für die Datenbankadministration und schließlich die Data Manipulation Language kurz DML für die Manipulation der Daten. Für die Anwendungsentwicklung mit relationalen Datenbanken sind hierbei die DDL und die DML wichtig, während die DCL eher dem Datenbankadministrator (DBA) vorbehalten bleibt. Jede SQL-Anweisung endet mit einem Semikolon, so daß SQL-Anweisungen über mehrere Zeilen und Abschnitte verteilt werden können. In den Anweisungen selbst ignoriert SQL die Groß- und Kleinschreibung, allerdings ist darauf hinzuweisen, daß bei der Groß- und Kleinschreibung der Daten selbst sehr wohl unterschieden wird.

Das SQL in Oracle besitzt folgende grundlegende Datentypen: für die Darstellung von Zahlen gibt es drei Datentypen, number für allgemeine Gleitkommazahlen, decimal bzw. integer für Festkommazahlen. Für die Darstellung von Zeichen stehen zwei grundlegende Datentypen zur Verfügung: char für Zeichenketten mit einer festen Länge für bis zu 255 Zeichen, varchar bzw. varchar2 für das Speichern von bis zu 2000 Zeichen variabler Länge. Mit Einschränkungen bei SQL Operationen wird von Oracle noch der Long Datentyp verwendet, sowie ab der Version 8 spezielle Datentypen, die sog. LOBs , die bis zu 4 Gigabyte Zeichen aufnehmen können. Datums- und Zeitinformationen werden in Oracle mit dem Datentyp date, normalerweise im Format TT-MM-JJ, verwaltet. Schließlich gibt es bei Oracle noch den Datentyp RAW und long RAW für die Speicherung von Binärdaten wie Bilder, Klänge und andere multimedialer Daten. Diese RAW Datentypen sind ebenfalls in ihrer Verwendung durch SQL Operationen eingeschränkt und können auch durch LOB's dargestellt werden. In den folgenden Abschnitten wird auf die wichtigsten PL/SQL Befehle der Oracle 8 DDL und DML eingegangen. Da der Umfang des von Oracle verwendeten PL/SQL sehr groß ist wird nur auf die wichtigsten und für diese Arbeit relevanten Anweisungen eingegangen. Um eine weitergehende Einführung in Oracle PL/SQL zu erhalten sei an dieser Stelle auf die im Anhang aufgeführte Literatur verwiesen.

1.3.2.1 Die Data Definition Language

Wenn man nun eine Datenbank mit SQL erstellen will bedient man sich der Datendefinitionssprache von SQL, um zu definieren, in welcher Form die abzubildenden Daten in der Datenbank abgelegt werden sollen. Hiermit werden die einzelnen Tabellen und ihre Struktur definiert, in dem später die Daten verwaltet werden sollen. Um nun eine Tabelle zu erstellen, verwendet man den create table Befehl. Hiermit wird eine Tabelle definiert, daß heißt es wird ein Name für die Tabelle angegeben, sowie die Spalten der Tabelle und die dazugehörigen Spaltennamen festgelegt. Weiterhin werden die Datentypen der Spalten, welche die Daten ja später aufnehmen sollen, angegeben. Informationen, welche die Konsistenz der Datenbank betreffen, wie Primärschlüssel, Fremdschlüssel bzw. ob ein Spalteninhalt Daten enthalten darf, die nicht definiert sind (NOT NULL) können innerhalb dieses Befehls ebenfalls festgelegt werden. Der Create table Befehl ist nach folgendem Muster aufgebaut:

```
CREATE TABLE taballenname (Spaltenname_1 datentyp [NOT NULL],
                                    ...     ...     ...,
                           Spaltenname_n datentyp [NOT NULL],
[CONSTRAINT regelname PRIMARY KEY (Spaltenname)],
[CONSTRAINT regelname FOTREIGN KEY (Spaltenname)
REFERENCES bezugstabellenname (Spaltenname*)];

*PK der Bezugstabelle
```

Mit dem create table Befehl wird also Tabelle definiert mit dem Namen tabellenname und den Spalte(n) Spaltenname_1 bis Spaltenname_n die einen Wert vom Typ datentyp aufnehmen können, also z.B. Zahlen, Buchstaben, Datum, Binärzeichen usw.

Mit dem optionalen Zusatz NOT NULL kann man Oracle anweisen in der entsprechenden Spalte keine nicht definierten Werte (Nullwerte) zu akzeptieren. Übrigens braucht der Zusatz NOT NULL nicht bei Primärschlüsselspalten angegeben werden, da der Primärschlüssel per Definition nicht undefiniert sein darf und Oracle hier automatisch keine nicht definierten Werte zuläßt. Optional ist es hier auch schon möglich den Primärschlüssel und Fremdschlüssel mit einer constraint Regel durchzusetzen. Wird ein Fremdschlüssel vergeben überprüft Oracle ob der angegebene Fremdschlüssel in der Datenbank als Primärschlüssel einer anderen Tabelle existiert.

Das Festlegen des Primärschlüssels und evtl. Fremdschlüssel kann aber auch später, nach der Definition der Tabelle festgelegt werden. Hier nun ein Beispiel für den Create Table Befehl:

```
CREATE TABLE Personal (Pers_nr NUMBER,
                       Vorname      VARCHAR2(50)      NOT NULL,
                       Nachname     VARCHAR2(50)      NOT NULL,
                       Eingestellt  DATE,
                       SozVersNr    VARCHAR2(30)      NOT NULL,
CONSTRAINT pk_personal PRIMARY KEY (Pers_nr));
```

Das Ergebnis dieses Befehls kann man sich als eine Tabelle mit dem Namen „Personal„ Vorstellen:

Pers_nr	**Vorname**	**Nachname**	**Eingestellt**	**SozVersNr**
NUMBER Primärschl.	VARCHAR2(5)	VARCHAR2(50)	DATE	VARCHAR2(30)

Um jetzt auch noch einen Fremdschlüssel zu definieren müssen wir erst einmal eine weitere Tabelle erstellen, auf die bezug genommen werden soll, und in der der Fremdschlüssel damit der Primärschlüssel ist:

```
CREATE TABLE SozVers (SozVersNr      VARCHAR2(30),
                      Arbeitsverh    VARCHAR2(15) NOT NULL,
                      AG_Ant_RE      NUMBER(6,2)
CONSTRAINT pk_sozvers PRIMARY KEY (SozVersNr));
```

Nun kann in der Personaltabelle ein Fremdschlüssel angelegt werden, der auf die SozVers Tabelle verweist:

```
TABLE Personal (Pers_nr,Vorname,Nachname,Eingestellt,SozVersNr)
CONSTRAINT fk_personal_sozvers
FOREIGN KEY (SozVersNr) REFERENCES SozVers (SozVersNr));
```

Es sei noch erwähnt daß man zusätzlich zur Definition des Primär- bzw. der Fremdschlüssel die Eindeutigkeit von Spalteninhalten durch die UNIQUE-Anweisung festlegen kann. Schließlich gibt es noch die Möglichkeit den Wertebereich der Spalten einzuschränken. Dieses kann man mit einer Check-Regel festlegen wobei auch noch die Möglichkeit der Angabe eines Standardwertes (Defaultwert) gegeben ist. Diese Check-Regel kann man z.B. bei der Spalte „Arbeitsverh„ in der Tabelle benutzen, um den möglichen Inhalt dieser Spalte auf die Werte „Arbeiter„ und „Angestellter„ zu begrenzen. Es kann im Laufe der Definition der Tabellen durchaus vorkommen, daß trotz eines gut geplanten Datenbankmodells, Änderungen an den Tabellen vorgenommen werden müssen. Hierfür benutzt man den Alter Table Befehl.

Es ist jedoch noch darauf hinzuweisen, daß sich keine Daten in den zu verändernden Tabellen befinden dürfen, und man muß bei der Änderung eines Datenbankmodells besonders die referenzielle Integrität der Datenbank beachten, wenn Primärschlüssel gelöscht bzw. verändert werden sollen. Löscht man nämlich einen Primärschlüssel einer Tabelle, welche in einer anderen Tabelle als Fremdschlüssel benutzt wird, so kann u.U. nicht mehr auf die Tabelle zugegriffen werden, weil der Verweis (Bezug) auf diese Tabelle nicht mehr vorhanden ist. Um einen Primärschlüssel zu verändern muß er erst mit dem drop Befehl gelöscht und dann mit dem alter table Befehl neu angelegt werden. (Der drop Befehl kann auch für das Löschen von ganzen Tabellen, Objekten oder Oracle Aliasdirectories verwendet werden.)

Den alter table kann man in drei Grundformen verwenden.

```
ALTER TABLE tebellenname {ADD|MODIFY}
                                (Spaltenname_1 Datentyp Integritätsregel,
                                ...,
                                 Spaltenname_n Datentyp Integritätsregel
```

Diese Form wird verwendet, um neue Spalten, den Primärschlüssel oder Fremdschlüssel in eine vorhandene Tabelle hinzuzufügen (Schlüsselwort ADD). Mit dem Schlüsselwort MODIFY verändert man eine vorhandene Spalte, also den Spaltenname oder den Datentyp bzw. dessen Länge (Spaltenbreite). Die dritte Form ermöglicht das Löschen von Integritätsregeln und den Primärschlüssel: Alter Table tabellenname Drop primary key. Mit den vorgestellten Anweisungen kann man also Tabellen und Integritätsregeln erstellen, löschen und verändern. Nun geht es darum die Tabellen mit Werten, also Daten zu füllen. Hierzu bedient man sich der Data Manipulation Language (DML) von SQL.

1.3.2.2. Die Data Manipulation Language

Die DML von SQL dient zum Verändern der Daten in den vorher definierten Tabellen. Mit ihr kann man Werte einfügen, löschen, ersetzen oder sich anzeigen lassen. Um sich Daten anzeigen zu lassen, gibt es in SQL den sehr umfangreichen select Befehl. Mit ihm gibt es so viele Möglichkeiten Daten aus einer oder mehreren Tabellen zu selektieren, daß hier nur die Grundlagen dieser Anweisung umrissen werden sollen. Eine einfache select Anweisung ist folgendermaßen aufgebaut:

```
SELECT Spaltenname_1 , ... , Spaltenname_n FROM Tabellennamen
```

Sie muß mit mindestens zwei Elementen beschreiben, den Spaltennamen bzw. die Liste der aufzurufenden Spalten einer Tabelle und den Namen der anzuzeigenden Tabelle. Der select Befehl geht die angegebene Tabelle durch und gibt die gewünschten Spalteninhalte (als temporäre Tabelle) aus. Einer solchen select Anweisung kann man mit Bedingungen für die Werte einer oder mehrerer Spalten ausführen lassen. Dieses geschieht mit Hilfe der where Klausel. Mit dieser where Klausel kann man auch mehrere Bedingungen mit dem logischen AND oder OR verknüpfen.

```
SELECT Spaltenname_1 , ... , Spaltenname_n FROM Tabellennamen
WHERE Spaltenname_x Bedingung ;
```

Die Ausgabe dieser neuen temporären Tabelle ist, da sie wie alle Tabellen in Oracle auf Relationen beruht nicht nach irgend einem Kriterium geordnet. Mann kann sich die select Ausgabe jedoch nach bestimmten Kriterien mit Hilfe der order by Klausel geordnet anzeigen lassen.

So kann man sich beispielsweise die Ausgabe eines select Befehls, der aus der Personaltabelle im vorigen Abschnitt alle Personen mit Vor- und Nachnamen ausgeben lassen deren Nachname mit W oder R beginnt und zwar in der Reihenfolge des Nachnamens alphabetisch geordnet.

```
SELECT Nachname,Vorname FROM Personal
WHERE Nachname like  'W%' OR Nachname like 'R%'
ORDER BY Nachname
```

Mit dem select Befehl kann man aber noch weiter Abfragen konstruieren. Zum Beispiel kann man in einer select Anweisung die Bedingung in der where Klausel als untergeordnete select Anweisung (Unterabfrage) angeben. Für das nächste Beispiel wird die SozVers Tabelle im vorherigen Abschnitt benutzt. Es sollen alle Sozialversicherungsnummern ausgegeben werden, wo der Arbeitgeberanteil an der Rentenversicherung kleiner oder gleich dem Durchschnitt aller in der Tabelle aufgeführten Arbeitgeberanteile ist.

```
SELECT SozVersNr FROM SozVers WHERE AG_Ant_RE <=
(SELECT avg (AG_Ant_RE) FROM SozVers);
```

Die Unterabfrage, das ist die Selectanweisung die in runden Klammern steht bildet zunächst den Durchschnitt aller Spalten AG_Ant_RE. Der Übergeordnete Selectbefehl gibt dann alle Sozialversicherungsnummern aus, die laut Bedingung kleiner oder gleich diesem Ergebnis sind.

Ich habe den Selectbefehl an den Anfang dieses Abschnitts gestellt, da er auch für die Modifizierung von Daten verwendet werden kann. Deshalb werden jetzt erst, im nächsten Unterabschnitt, die Befehle für das Hinzufügen, Löschen und Verändern von Tabellenwerten vorgestellt.

Als erstes sollen einer definierten Tabelle Werte hinzugefügt werden. Das macht man mit Hilfe der Insert Anweisung. Die Insert Anweisung hat folgende Form:

```
INSERT INTO Tabellenname
[(Spaltenname_1,...,Spaltenname_n)]
VALUES (Wert für Spalte_1,...,Wert für Spalte_n);
```

Man könnte z.B. in die Tabelle Personal die folgenden Werte eingeben:

```
INSERT INTO Personal
(Pers_nr,Vorname,Nachname,Eingestellt,SozVersNr)
VALUES ('105831','Herbert','Feuerstein','03-Jun-98');
```

Bei den Wertangaben ist darauf zu achten, daß die Anzahl der Werte, die Reihenfolge und die Datentypen mit den Spaltendefinitionen der betreffenden Tabelle übereinstimmen. Würde man z.B. im obigen Beispiel die Spalte Pers_nr mit dem Wert ‘einhundertfünftausendachthunderteinunddreißig‘ angeben, würde das zu einer Fehlermeldung führen, da der angegebene Wert keine Ziffernfolge (number) ist. Mit dieser Anweisung kann man nun Zeile für Zeile Werte in die Tabelle schreiben. Der insert Befehl kann auch mit einer Select Unterabfrage verwendet werden. Dieses ist aber nur dann möglich wenn die Daten bereits in irgend welchen anderen Tabellen in der Datenbank gespeichert sind. Eine weitere Möglichkeit der Datenmanipulation ist das Verändern bereits gespeicherter Werte in den Tabellen. Dieses kann mit dem update Befehl erfolgen. Der update Befehl sieht folgendermaßen aus:

```
UPDATE Tabellenname SET
Spaltenname_1=Werte_1,...,Spaltenname_n=Wert_n
[WHERE Bedingung];
```

Anders als bei der insert Anweisung werden beim Update Befehl ohne Angabe einer Where Klausel alle vorhandenen Zeilen verändert. Um bestimmte Zeilen zu verändern muß die where Klausel mit einer Bedingung angegeben werden:

```
UPDATE Personal SET
Nachname = ‘Müller‘ WHERE Pers_Nr = ‘105831‘;
```

Hierbei wird nur in der Zeile der Spalteneintrag Nachname zu ‘Müller‘ verändert, in der der Spalteneintrag Pers_Nr gleich 105831 ist.

Als letzte Möglichkeit der Spatenmanipulation soll noch die Delete Anweisung erwähnt werden. Mit ihr werden Werte aus einer Tabelle gelöscht. Sie hat folgendes Aussehen:

```
DELETE FROM Tabellenname [WHERE Bedingung];
```

Hierbei werden alle Zeilen gelöscht, die die in der where Klausel angegebene Bedingung erfüllen. Beim Löschen ganzer Tabelleninhalte kann man auch die schnellere truncate Table Anweisung benutzen. Hierbei ist allerdings darauf zu achten, daß sie keine echte DML Anweisung ist und nicht z.B. durch ein rollback rückgängig gemacht werden kann. Der letzte Abschnitt sollte einen groben Überblick über die wichtigsten SQL Anweisungen geben, um den Gebrauch des PL/SQL cartridge im WAS bzw. das Programmbeispiel im zweiten Teil dieser Arbeit verständlich zu machen.

Oracle SQL bietet noch wesentlich mehr Anweisungen, Abwandlungen der hier vorgestellten Anweisungen, wie integrierte Funktionen für numerische Werte und Zeitdatentypen, Selectanweisungen mit deren Hilfe Joins über mehrere Tabellen ausgeführt werden, verschiedene Datensichten usw. Da dieses nicht für das Verständnis des zweiten Teils der Arbeit unbedingt erforderlich ist komme ich jetzt zu den Oracle Erweiterungen, dem PL/SQL von Oracle, da PL/SQL im Beispiel verwendet wird. Das PL/SQL kombiniert die nicht prozedurale Sprache SQL mit Elementen von strukturierten Prozeduralen Sprachen, wie beispielsweise C. Hierbei kann man z.B. Schleifen und Verzweigungen Programmieren, Prozeduren und Funktionen benutzen und aufrufen lassen. Seit der Version 8 von Oracle ist es auch möglich, eigene Datentypen wie Records zu erstellen, was eine Annäherung an objektorientierten Programmiersprachen erkennen läßt.

1.3.3. PL/SQL Erweiterung von Oracle

Grundlegend handelt es sich bei PL/SQL um eine blockorientierte Sprache, welche pro Block aus drei Abschnitten bestehen kann. Als erstes den Deklarationsabschnitt, dann den ausführbaren Code und schließlich den Exceptionabschnitt. Ein PL/SQL Block besteht, ähnlich wie bei anderen Programmiersprachen wie z.B. C, aus Prozeduren und Funktionen aber kann auch ein sog. Anonymer PL/SQL Block sein. Ein solcher anonymer PL/SQL Block bekommt keinen Namen und keine Argumente –im Gegensatz zu Prozeduren und Funktionen- und kann auch keine Werte, wie bei Funktionen, zurückgeben. Ein anonymer PL/SQL Block eignet sich daher nur für Skripte, die direkt aufgerufen werden müssen, also nicht aus einem anderen Programm heraus aufgerufen werden können. Anonyme PL/SQL Programmblöcke werden auch nicht in der Datenbank selbst abgelegt. Daher ist es für die Anwendungsentwicklung besser, Programme in Form von Prozeduren und Funktionen zu schreiben, da sie von einem beliebigen PL/SQL Unterprogramm aufgerufen werden können und deshalb wiederverwendbar sind. Außerdem lassen sie sich in sog. Paketen zusammenfassen. Das Programmbeispiel im zweiten Teil dieser Arbeit besteht daher nicht aus anonymen Blöcken, sondern nur aus Prozeduren und Funktionen.

1.3.3.1 Deklaration von Variablen in PL/SQL

Zunächst einmal können alle von Oracle SQL unterstützten Datentypen auch als Variablen im Deklarationsteil eines PL/SQL Blocks deklariert werden. Zusätzlich besteht die Möglichkeit weitere Datentypen, die nur von PL/SQL unterstützt werden als Variablen zu deklarieren. Dies sind boolsche Variablen (boolean), welche die Werte true, false und auch den Wert Null (nicht definiert) annehmen können. Weiterhin gibt es den Datentyp binary integer für vorzeichenbehaftete ganze Zahlen im Wertebereich von –2147483647 bis +2147483647, die Datentypen natural und positiv verwenden Untermengen dieses Wertebereichs von 0 bis +2147483647 bzw. von +1 bis +2147483647.

Weiterhin können in PL/SQL Variabledeklarationen mit dem Schlüsselwort %type deklariert werden, wobei einer Variable der gleiche Datentyp zugewiesen wird, wie der einer bereits deklarierte Tabellenspalte. Die %type Deklaration sieht folgendermaßen aus:

Variabelnamen Tabellennamen.Spaltennamen%type

Dieses führt zu einer leichteren Wartbarkeit den PL/SQL Programmcodes, da sich bei einer Änderung der Datentypdeklaration der betreffenden Spalte die abhängigen Variablen automatisch anpassen. Schließlich lassen sich sogar ganze records abhängig von einer Tabellenzeile deklarieren. Dieses geschieht mit der %rowtype Anweisung, die ähnlich wie die %type Anweisung funktioniert. Ab der Version 8 von Oracle gibt es weitere Datentypen, wie beispielsweise sog. LOBs, die große Datenmengen aufnehmen können oder selbst definierte Records und Objekte. Diese neuen Datentypen möchte ich aber am erst am Schluß dieses Kapitels über PL/SQL kurz vorstellen.

1.3.3.2. PL/SQL Programmierungskonstrukte

Der Programmablauf in PL/SQL wird von folgenden Grundlegenden Programmstrukturen gesteuert: Es gibt die if-then-else Anweisung für Verzweigungen im Programm. Sie funktioniert größtenteils so wie bei vergleichbaren Programmiersprachen. Hierbei ist jedoch zu beachten, daß die elsif und else Klauseln optional sind, es können mehrere elsif Anweisung in einem if Abschnitt stehen, jedoch nur ein else. Außerdem wird die elsif Anweisung ohne e geschrieben, also nicht etwa elseif und ein if Abschnitt wird immer mit END IF abgeschlossen. Ein weiteres Programmkonstrukt ist die Schleife. Befehle, die zwischen einer LOOP und END LOOP Anweisung stehen werden als Schleife immer wieder durchlaufen. Zum Verlassen der Schleife wird zwischen LOOP und END LOOP eine EXIT Anweisung eingebaut. Die EXIT Anweisung kann ohne eine Bedingung als unbedingter Abbruch der Schleife programmiert werden oder mit einer Bedingung (EXIT WHEN Bedingung) verlassen werden, wenn die Bedingung erfüllt ist. Eine zweite Schleifenkonstruktion ist die WHILE...LOOP Schleife. Dieses ist eine kopfgesteuerte Schleife in der Form WHILE Bedingung LOOP ... END LOOP. Diese Schleife wird nur so lange ausgeführt, wie die Bedingung im Schleifenkopf erfüllt ist. Sind die Schleifendurchgänge vor dem Start einer Schleife bekannt, also nicht von einer Bedingung abhängig, die bei irgend einem Schleifendurchlauf eintritt, benutzt man in PL/SQL die FOR-LOOP Schleife. Die Form ist:

```
FOR Schleifenvariable IN [REVERSE] untereGrenze..obereGrenze
LOOP
.......;
END LOOP;
```

Hierbei ist die Schleifenvariable eine Integervariable, die als Zähler die Schleifendurchläufe zählt. Sie wird auf einen Wert gesetzt, bei dem sie anfängt zu zählen. Bei jedem Schleifendurchgang wird sie um eins inkrementiert(bzw. wenn das Schlüsselwort REVERSE benutzt wird dekrementiert, wobei man natürlich obere und untere Grenze vertauschen muß) bis die angegebene obere Grenze erreicht ist – die Schleife wird dann verlassen.

Hier ein Beispiel für solch eine FOR-LOOP Schleife:

```
FOR i IN 1..100 LOOP
        x=x+y;
END LOOP;
```

Die Zählervariable i startet mit dem Wert 1. Bei jedem Schleifendurchlauf wird sie um eins erhöht, bis sie den Wert 101 annimmt. Hat sie den Wert 100 erreicht wird die Schleife also ein letztes Mal durchlaufen es werden also insgesamt 100 Schleifendurchläufe ausgeführt. In PL/SQL ist es auch möglich durch goto Anweisungen im Programm an eine beliebige Codestelle, die durch ein sog. Label gekennzeichnet wird, zu springen. Im Programmcode dieser Arbeit wird dieser Befehl allerdings nicht benutzt. Ansonsten werden in PL/SQL die Operatoren, wie z.B. := als Zuweisung, <,>,=,<=,>= ähnlich wie bei anderen bekannten Programmiersprachen verwendet. Auf eine Besonderheit ist jedoch bei Nullwerten hinzuweisen. Eine Überprüfung, ob eine bestimmte Variable Null, also nicht definiert ist, wird nicht durch den = oder != Operator überprüft, sondern durch IS NULL oder IS NOT NULL. Abschließend sei noch erwähnt, daß Kommentare im PL/SQL Code, wie bei C mit den Zeichen /* Kommentar */ umschlossen werden können. Zeilenweise können Kommentare auch mit - - begonnen werden.

1.3.3.3. Integrierte Funktionen für Zeichenketten

Das Beispielprogramm am Ende dieser Arbeit verwaltet und bearbeitet viele Zeichenketten und benutzt daher einige der von Oracle zur Verfügung gestellten Funktionen für die Verwaltung und Manipulation von Zeichenketten. Aus diesem Grund möchte ich an dieser Stelle einige der wichtigsten Zeichenkettenfunktionen von Oracle beschreiben. Für viele Funktionen ist die Angabe der Länge einer Zeichenkette wichtig. Hierzu gibt es die Funktion LENGTH(Zeichenkettennamen). Die so ermittelte Länge einer Zeichenkette kann von anderen Funktionen für die Manipulation derselben benutzt werden. So gibt es die Möglichkeit, eine Teilzeichenfolge aus einer Zeichenkette zu gewinnen. Die Funktion:

SUBSTRING(name,start,anzahl);

Erlaubt es aus der Zeichenketten „name“ von dem Zeichen start an (start ist das n.te Zeichen ab dem die Zeichenteilfolge gebildet werden soll) so viele Zeichen auszulesen, wie in anzahl angegeben. Wen z.B. die Zeichenkette name den Inhalt „Inhalt“ hat, würde die Funktion SUBSTRING(name,3,4) die Zeichenkette „halt“ zurückliefern. Eine weitere Funktion, die im Programm benutzt wird ist die

REPLACE(name,‘zeichenfolge‘,‘neue zeichenfolge‘);

Funktion. Hierbei wird aus der Zeichenkette name die Zeichenfolge ‘zeichenfolge‘ durch die Zeichenfolge ‘neue zeichenfolge‘ ersetzt. Wird der letzte Parameter nicht angegeben, so wird die Zeichenfolge einfach gelöscht. Z.B.: REPLACE(name,‘In‘,‘An‘); - macht aus der Zeichenfolge „Inhalt“ die Zeichenfolge „Anhalt“.

Eine weitere Zeichenkettenoperation, die im Beispielprogramm verwendet wird, ist die Funktion:

```
INSTR(‘Zeichenkette‘,‘Suchwort‘);
```

Diese Funktion gibt die Position des ersten Zeichens einer Teilzeichenkette in einer Zeichenkette zurück:

```
i := INSTR(‘Zeichenkette‘,‘enk‘);
```

Setzt die Variable i auf den Wert 6, da das „e“ von „enk“ in „Zeich**enk**ette“ an der 6. Stelle gefunden wird. Diese Funktion gibt es auch für LOBs (dbms_lob.instr) und funktioniert hier genau so.

1.3.3.4. Deklaration von Prozeduren und Funktionen

Im Gegensatz zu anonymen Blöcken werden Prozeduren und Funktionen in der Datenbank abgelegt, d.h. sie sind permanent und können so von beliebigen Programmen aufgerufen werden. Eine Prozedur in PL/SQL ist folgendermaßen gegliedert:

```
CREATE [OR REPLACE] PROCEDURE
 Prozedurname [(argument_1,...,argument_n)] IS
[lokale Variabeldeklarationen]
BEGIN
Ausführungsteil
[Exeptionteil]
END [Prozedurname];
```

Der Prozedurname unterliegt den Einschränkungen der Namensgebung von Objekten in Oracle. Optional ist die Deklaration von Argumenten, die beim Prozeduraufruf übergeben werden. In den lokalen Variabledeklarationen werden die in der Prozedur verwendeten Variablen festgelegt. Danach folgt der Ausführungsteil, der aus PL/SQL Anweisungen besteht. Schließlich können im optionalen Exeptionteil Fehler abgefangen werden. Das Schlüsselwort END schließt die Prozedur ab. Eine Funktion ist ähnlich wie eine Prozedur aufgebaut, der Unterschied besteht darin, daß eine Funktion immer mindestens einen Wert an die Aufrufende Instanz zurückgibt. Der generelle Aufbau einer Funktion ist folgendermaßen:

```
CREATE [OR REPLACE] FUNCTION
Funktionsname [(argument_1,...,argument_n)]
RETURN Rückgabedatentyp IS
[lokale Variablendeklarationen]
BEGIN
Ausführungsteil
[Exeptionteil]
END [Funktionsname];
```

Der Datentyp, der zurückgegeben wird, wird nach dem Schlüsselwort RETURN angegeben. Gespeicherte Prozeduren und Funktionen können vom Benutzer direkt aufgerufen werden, oder von anderen PL/SQL Programmen aufgerufen werden, es ist sogar möglich, daß sich Prozeduren und Funktionen selber aufrufen (Rekursivität). Beim Aufruf von Prozeduren und Funktionen in einem PL/SQL Unterprogramm muß die aufzurufende Prozedur oder Funktion immer deklariert sein, d.h. der Datenbank bekannt sein. Hierdurch ergibt sich die Notwendigkeit der Vorwärtsdeklaration bei Prozeduren oder Funktionen, die sich gegenseitig aufrufen.

1.3.3.5. Auslesen von Tabellendaten mit Hilfe von Cursors

Für das Abrufen von Daten aus SQL Tabellen von Oracle bzw. das Verwalten des hierzu notwendigen Select Befehls verwendet PL/SQL sog. Cursors. Cursors sind deswegen notwendig, da es in PL/SQL Programmen nicht vorgesehen ist, daß das Ergebnis eines Select Befehls aus mehr als einer Zeile besteht. Einen Cursor verwendet man in vier Schritten. Als erstes muß der Cursor deklariert werden, dann wird er für das Auslesen geöffnet, jetzt kann mit Hilfe des Cursors eine Tabelle ausgelesen werden und zum Schluß wird der Cursor wieder geschlossen. Die Cursordeklaration erfolgt im Deklarationsteil eines PL/SQL Programms. Die Deklaration sieht ähnlich aus wie eine ganz normale Variablendeklaration.

```
CURSOR Cursorname IS SELECT-Befehl;
```

Man kann sich unter einem Cursor eine Art Schlüsselname vorstellen, hinter dem sich eine komplette Selectanweisung verbirgt. Das besondere hieran ist jedoch das diese Selectanweisung nicht auf der ganzen Tabelle ausgeführt wird, sonder innerhalb der Tabelle zeilenweise ausgeführt wird, wie später noch erklärt wird. Bevor der Cursor benutzt werden kann wird er, am besten ganz am Anfang des Ausführungsteils eines PL/SQL Programms geöffnet. Dies geschieht durch die Anweisung :

```
OPEN Cursorname;
```

In einem PL/SQL Programm können mehrere Cursors deklariert werden und auch gleichzeitig geöffnet werden. Es ist allerdings darauf zu achten, daß nach dem Benutzen eines Cursors, spätestens jedoch am Ende des Anweisungsteils die Cursors auch wieder geschlossen werden. Dieses geschieht mit dem CLOSE Cursorname Befehl. Ein Cursor gibt beim Lesen von Tabellenwerten Zeile für Zeile, entsprechend dem sich hinter dem Cursor verbergenden Selectbefehl, Daten aus. Diese müssen dann in dem der Tabelle verwendeten Datentypen entsprechenden Variablen eingelesen werden. Um eine ganze Tabelle auszulesen wird der Auslesebefehl eines Cursors innerhalb einer Schleife ausgeführt. Man verwendet hierzu eine Loop-Schleife, die verlassen wird, wenn die Tabelle ausgelesen ist, der Cursor also die letzte Zeile der Tabelle, in der sich Daten befinden ausgelesen hat. Findet der Cursor am Ende der Tabelle keine Daten mehr gibt er einen bestimmten Wert zurück. Diesen Wert kann man mit der %notfound Anweisung abrufen. Gibt der Cursor diesen Wert aus kann die Schleife verlassen werden.

Das eigentliche Auslesen und abspeichern der Tabellenwerte in Variablen geschieht mit der FETCH Anweisung. Hier ein kurzes Beispiel für die Verwendung eines Cursors. Es sollen aus der SozVers Tabelle (Kap. 1.3.2.1.) alle Arbeitgeberanteile Rentenversicherung (AG_ANT_RE) aufsummiert werden. Hierzu wird die Spalte AG_Ant_RE Zeile für Zeile ausgelesen und in der Variable sum gespeichert und dann in die Variable summe aufsummiert.

```
1     CREATE OR REPLACE FUNCTION Aufsummieren
2     RETURN NUMBER (6,2) IS
3     summe SozVers.AG_Ant_RE%type :=0.00;
4     sum     SozVers.AG_Ant_RE%type :=0.00;
5     CURSOR get_AGAnteil IS SELECT AG_Ant_RE FROM SozVers;
6     BEGIN
7       OPEN get_AGAnteil;
8       LOOP
9         FETCH get_AGAnteil INTO sum;
10        EXIT WHEN get_AGAnteil%notfound;
11        summe := summe + sum;
12      END LOOP;
13      CLOSE get_AGAnteil;
14      RETURN summe;
15    END Aufsummieren;
```

Ich habe in diesem Beispiel die Zeilen mit Zeilennummern versehen, um die einzelnen Schritte besser erklären zu können. In Zeile 1 wird die Funktion Aufsummieren deklariert. Sie liefert einen Rückgabewert (Zeile 2) vom Datentyp NUMBER mit insgesamt 6 Stellen inklusive 2 Nachkommastellen zurück. Dieser Datentyp entspricht dem der Spalte AG_Ant_RE in der Tabelle SozVers. In Zeile 3 und 4 werden 2 Variablen (summe und sum) deklariert, die vom gleichen Datentyp sind, wie die Spalte AG_Ant_RE der Tabelle SozVers also NUMBER (6,2).

Die Variablen werden auf den Wert 0,00 gesetzt. In Zeile 5 wird ein Cursor get_AGAnteil deklariert mit einem Selectbefehl, der die Werte der Zeile AG_Ant_RE ausgeben soll. Nun beginnt an Zeile 6 der Ausführungsteil des Programms (Schlüsselwort BEGIN). In Zeile 7 wird der Cursor geöffnet. Ab Zeile 8 beginnt die Schleife. In Zeile 9 wird die erste Zeile der Tabelle ausgelesen, also der Wert, der in der ersten Zeile der Tabelle in der Spalte AG_Ant_RE steht wird in die Variable sum eingelesen. In Zeile 10 wird überprüft, ob der Cursor den Wert %notfound ausgibt, er also keinen Wert gefunden hat. Ist dieses der Fall wird die Schleife verlassen, d.h. der Code wird ab Zeile 13 weiterverarbeitet. In Zeile 11 wird bei jedem Schleifendurchlauf die Variable summe um den Wert der Variablen sum erhöht, also aufsummiert. Zeile 12 schließt die Schleife ab und der Code wird am Schleifenanfang, also ab Zeile 9 wiederholt, nur das der Cursor sich bei der nächsten FETCH Anweisung die nächste Tabellenzeile zum auslesen vornimmt. Findet der Cursor keine Werte mehr wird durch die EXIT Anweisung die Schleife verlassen und die Programmausführung ab Zeile 13 weitergeführt. In Zeile 13 wird der Cursor geschlossen. Zeile 14 gibt das aufsummierte Ergebnis dieser kleinen Funktion an das aufrufende PL/SQL Programm zurück. Zeile 15 beendet die Funktion.

1.3.3.6. Neue PL/SQL Datentypen

Ab der Version 8 von Oracle ist die Verwendung von neuen Datentypen möglich. Dies sind teilweise Erweiterungen bestehender Datentypen, wie die bekannten RAW Datentypen, aber auch neue Datentypen, die aus neuen Konzepten, wie objektorientierte Ansätze, resultieren. Als Erweiterung des RAW bzw. long RAW Datentyps, also für die Speicherung größerer Datenmengen wurden neue Datentypen, sog. LOBs (Large Objects) eingeführt. Mit diesen Datentypen können Datenmengen bis zu 4 Gigabytes verwaltet werden (long RAWs hatten eine Kapazität von nur 2 Gigabyte). Die LOBs unterteilen sich in vier Datentypen. Als erstes in die BLOBs oder Binary Large Objects für die Verwaltung von Binärdaten. Dieses können beispielsweise Grafikdateien, Tondateien oder irgend welche andere Multimediadateien sein. Für Textdateien verwendet man CLOBs und NCLOBs. NCLOBs sind Zeichenfolgedaten mit länderspezifischen Zeichensätzen. Schließlich gibt es noch den Datentyp BFILE. Dieser Datentyp ist im Gegensatz zu den anderen LOBs ein sog. externer LOB. Mit ihm ist es möglich, externe Dateien zu verwalten. Extern bedeutet Dateien, die sich nicht in der Datenbank befinden, sondern vom Betriebssystem verwaltet werden. Also z.B. eine mit einem Texteditor erstellten Textdatei, welche sich unter einem Pfad des Betriebssystems befindet. Mit diesen BFILE Datentyp ist es möglich Daten aus einer Datei in die Datenbank hereinzunehmen d.h. eine Datei auszulesen und in einen internen LOB zu speichern. Solche internen LOB's können ausgelesen werden und die Daten in einem PL/SQL Programm weiterverwendet werden. Mit den BFILE ist es ebenfalls möglich Daten auf die Betriebssystemebene zurückzuschreiben.

Der BFILE Datentyp kann als ein Zeiger bzw. Adresse auf eine Datei aufgefaßt werden, die sich unter irgend einem Pfad der betriebssystemverwalteten Dateiordnung befindet. Ein BFILE besteht aus zwei Komponenten. Einmal aus dem Dateinamen und zum anderen aus einem Verzeichnisalias, welches den Pfad der entsprechenden Datei beschreibt. Um BFILES benutzen zu können muß der Datenbank der physikalische Pfad, in dem sich die Datei befindet bekannt gemacht werden. Dieses geschieht mit dem CREATE DIRECTORY Befehl. Möchte man z.B. eine Datei in die Datenbank einlesen, welche sich unter dem Pfad C:\Dateien\Bilder\gifs\ befindet legt man als erstes den Verzeichnisalias fest:

```
CREATE DIRECTORY "myname" AS 'C:\Dateien\Bilder\gifs';
```

Hiermit wird dem Datenbanksystem der Zugriff auf das Verzeichnis gewährt und für dieses Verzeichnis der Alliasname myname vergeben. Ein BFILE wird genauso wie die anderen LOB's und Variablen im Deklarationsteil eines PL/SQL Programms deklariert. Er erhält also einen Namen und als Datentyp BFILE. Im Gegensatz zu internen LOB's müssen BFILES vor ihrer Verwendung nicht initialisiert werden. Nach dem Einrichten eines Verzeichnisalias und der Deklaration des BFILE Datentyps für eine Tabellenspalte kann der Verzeichnisalias und der Dateiname mit dem SQL Befehl insert eingefügt werden. Ein Beispiel für das Speichern von Werten in einer Tabellenspalte mit dem Datentyp BFILE könnte sein :

```
INSERT INTO Tabellenname VALUES
(...,bfilename('Verzeichnisalias','datei.name'),...);
```

1.3.3.6.1. Probleme beim Einlesen von externen Dateien mit Hilfe von BFILEs

Wie bereits beschrieben, muß der Betriebssystempfad der Datei, die mit Hilfe eines Bfiles eingelesen werden, soll mit dem Create Directory Befehl der Datenbank bekannt gemacht werden bzw. die Erlaubnis zum Lesen von Dateien in diesem Verzeichnis dem Datenbankbenutzer gegeben werden. Der Create Directory Befehl wird vor der Ausführung des Programms, welches die Dateien auslesen soll, entweder direkt in SQL eingegeben, oder als anonymer Block ausgeführt. Dieses hat aber den großen Nachteil, daß die Directories, in denen sich die auszulesenden Dateien befinden, vor dem Auslesen bekannt sein müssen, da im Programm selbst eine Änderung des Pfades nicht ohne weiteres möglich ist, d.h. im Programm selbst kann dem Aliasdirectory kein anderer physikalischer Pfad mehr zugewiesen werden. Diese Unflexibilität des Create Directory Befehls ist vor allem dann unvorteilhaft, wenn vor der Programmausführung die Dateien, bzw. Pfade der Dateien noch nicht feststehen oder viele Dateien aus ganz unterschiedlichen Pfaden ausgelesen werden sollen. Bei der Entwicklung des Beispielprogramms (Suchmaschine) in dieser Arbeit müssen aber sehr viele Dateien aus vielen unterschiedlichen Verzeichnissen zwecks Vergleich mit einem Suchwort eingelesen werden. Um das Problem zu lösen, habe ich im Programm, welches die Dateien auslesen soll Veränderungen an einer Oracle Systemtabelle vorgenommen. Die internen Informationen einer relationalen Datenbank werden ebenfalls als Relation, in einer Tabelle, abgelegt. Viele dieser Systemtabellen befinden sich bei Oracle 8 unter dem Benutzer SYS.

Die Systemtabellennamen enden dort alle mit einem $-Symbol. Die Betriebssystempfade, die Oracle mit dem Create Directory Befehl einen Verzeichnisalias zuweist befinden sich im SYS Account der Datenbank in der Systemtabelle DIR$ unter der Spalte OS_PATH. Um nun einem Verzeichnisalias bei Ausführung eines PL/SQL Programms verschiedene Betriebssystempfade zuzuweisen, kann man den Eintrag in der Systemtabelle DIR$ in der Spalte OS_PATH ändern. Es ist jedoch wichtig, daß Änderungen an diesen Tabellen nur vorgenommen werden können, wenn ein UPDATE Recht auf diese Tabelle besteht (Dieses kann man als SYSDBA vom Security Manager der Datenbank dem entsprechenden Account zuweisen). Mit dem Befehl:

UPDATE SYS.DIR$ SET OS_PATH = verzname WHERE OS_PATH = altname;

kann man den Betriebssystempfad eines Aliasdirectories im Programm ändern, und dann die Datei ganz normal mit einem Bfile auslesen. Man sollte allerdings darauf achten, daß am Ende der Prozedur der Orginalzustand der Systemtabelle wieder hergestellt wird. (Vor Änderung des Eintrags Orginalzustand speichern und am Ende der Prozedur wieder mit UPDATE zurückschreiben).

1.3.3.6.2. Funktionen für LOBs

Oracle stellt für die Verwaltung von LOB's ein spezielles Paket zur Verfügung, das sog. dbms_lob Paket. Speziell für BFILES gibt es Funktionen für das Öffnen einer Datei (dbms_lob.FILEOPEN()), für das Prüfen ob eine Datei geöffnet ist (dbms_lob.FILEISOPEN()), für das Schließen einer geöffneten Datei (dmbs_lob.FILECLOSE)) oder aller geöffneten Dateien (dbms_lob.FILECLOSEALL()), sowie für das Nachprüfen ob eine Datei mit einem bestimmten Namen im angegebenen BFILE existiert (dbms_lob.FILEEXISTS()). Um den Inhalt den BFILE Datentyps auszulesen gibt es die Funktion dbms_lob.FILEGETNAME(). Dieser Inhalt stellt allerdings nicht den Inhalt der Datei dar sondern ist ein Verweis auf eine Datei. Um nun auf den Inhalt einer Datei zugreifen zu können muß der Inhalt der Datei auf die ein BFILE verweist, in ein internes LOB kopiert werden. Im Unterschied zu dem externen Datentyp BFILE muß ein internes LOB vor dem ersten Gebrauch initialisiert werden. Diese Initialisierung kann nicht direkt z.B. mit dem Zuweisungsoperator erfolgen. Um die Initialisierung zu erleichtern, sollte eine öffentlich zugängliche Tabelle in der Datenbank aufgenommen werden. Die Tabellendeklaration sollte folgendermaßen aussehen:

```
CREATE TABLE temp_blob (
temp_blob      BLOB,
temp_clob      CLOB,
temp_nclob     NCLOB);
```

Im zweiten Schritt wird die Tabelle mit einer Zeile initialisierter LOB's gefüllt:

```
INSERT INTO temp_blob VALUES (
empty_blob(), empty_clob(), empty_nclob());
```

Nun kann der Datenbankprogrammierer mit einer einfachen SELECT Anweisung seine eigenen LOB's mit Hilfe dieser Tabelle initialisieren:

```
SELECT temp_blob INTO myblob FROM temp_blob FOR
UPDATE;
```

Das Verändern von LOB Werten wird derzeit nicht von der automatischen Locking Funktion durch Oracle unterstützt. Deshalb ist der Zusatz FOR UPDATE notwendiger Weise bei der Initialisierung mit anzugeben. Wird dieser Zusatz nicht mit angegeben entsteht bei der späteren Zuweisung von Werte in diese LOB's eine Oraclefehlermeldung ORA22920 „Zeile mit dem LOB Wert ist nicht gesperrt„. Auch um diese internen LOB's verwenden zu können stellt das dbms_lob Paket Prozeduren und Funktionen zur Verfügung. Die wichtigsten sind: dbms_lob.COPY() zum Kopieren eines Teils oder den gesamten LOB Inhalt, dbms_lob.ERASE() zum Löschen des Inhalts, dbms_lob.WRITE() für das Schreiben in ein LOB und schließlich für das Kopieren des Inhalts eines BFILE in ein internes LOB die Funktion dbms_lob.LOADFROMFILE(). Mit dbms_lob.GETLENGTH() erhält man die Länge eines LOBs. Eine weitere wichtige Funktion ist die dbms_lob.INSTR() Funktion. Hiermit kann ein LOB nach einer angegebenen Zeichenkette bzw. Muster durchsucht werden. Dieses kann an einer beliebigen Stelle (Offset) geschehen und das Suchmuster kann bis zu 32767 Bytes groß sein.

Im letzten Abschnitt dieses Kapitels möchte ich noch kurz auf die objektorientierten Ansätze seit der Oracle 8 Version eingehen. Oracle stellt hierzu benutzerdefinierbare Datentypen, die sog. records zur Verfügung. Darüber hinaus kennt Oracle sog. Collection Datentypen. Diese Collections sind VARRAYS und NEESTED TABLES (Verschachtelte Tabellen). Records können vom Benutzer selbst definiert werden. Dieses geschieht mit der Deklaration in der Form:

```
CREATE TYPE mytype AS OBJECT;
```

Der Umgang mit diesen Datentyp ist in weiten Teilen ähnlich wie in anderen objektorientierte Programmiersprachen. Es gibt Methoden, die diesen Objekten zugeordnet sind und auf die nur diese Objekte zugriff haben (Kapslung) und ebenso gibt es Konstruktoren für diese records, die allerdings erst beim Speichern von Werten in diese Records mit anzugeben sind. Auf eine vollständige Erklärung des objektorientierten Ansatzes möchte ich aber aus Platzgründen und auch von der Themenstellung dieser Arbeit her verzichten. Außerdem sind sie für das Verständnis dieser Arbeit nicht weiter wichtig.

Bis zu dieser Stelle habe ich die Funktionen des Internets sowie von PL/SQL vorgestellt, da beide Gebiete die Hauptschnittstellen des Web Application Servers darstellen. Im nächsten Kapitel kann ich nun endlich auf den Web Application Server selbst eingehen.

2. Aufbau und Funktion des WAS

2.1. Aufbau des Web Application Servers

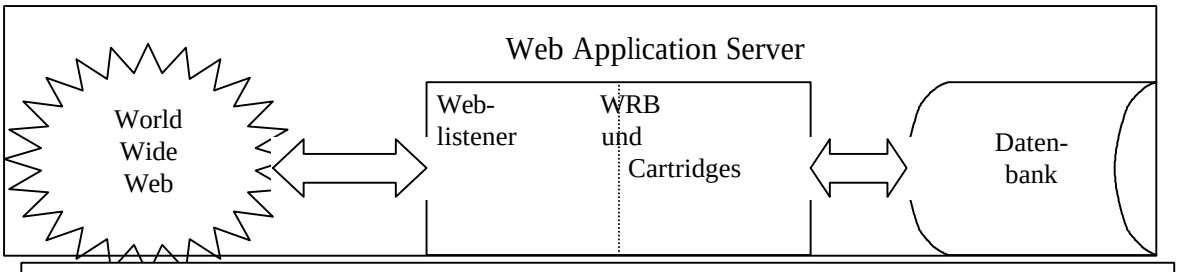


Bild 2 : Übersicht WAS

Der Web Application Server setzt sich aus einer Vielzahl von Prozessen zusammen, die auf einem oder auch mehreren Rechnern ablaufen. Man kann den Web Application Server in drei Hauptprozesse unterteilen. Als erstes den Weblistener oder auch Webdaemon genannt als Schnittstelle zum WWW. Er ist im Web Application Server Paket enthalten, es ist aber auch möglich fremde Weblistener mit dem Web Application Server einzusetzen. Der Web Application Server benutzt in seiner Standardinstallation mindestens zwei Weblistener an zwei unterschiedlichen Ports. Einen sogenannten Administrativlistener für die Administration des Web Application Servers über das WWW, sowie einen zweiten Weblistener für das Verwalten der mit dem Web Application Server erzeugten Webseiten bzw. deren Bereitstellung im Web. Es ist allerdings möglich, weitere Webdaemons an anderen Ports einzurichten und für eine Web Application Server Installation zu benutzen. Die zweite Hauptkomponente des Web Application Servers sind die sog. Cartridges. Dieses sind ausführbare Programme, welche die Schnittstelle zu den vom Anwendungsentwickler geschriebenen Programmen bilden.

Jedes Cartridge steht für eine andere Programmiersprache. Die Standardcartridges, die im Web Application Server Paket enthalten sind, können für die Sprachen PL/SQL, Perl und Java eingesetzt werden. Es gibt weitere Cartridges für das Anstoßen anderer Programme, sowie die Möglichkeit eigene Benutzercartridges zu schreiben. Die dritte und wichtigste Komponente des Web Application Servers ist der Web Request Broker oder WRB. Der WRB hat die Aufgabe Anfragen vom Internet zu verwalten. Er verteilt die Anfragen an die anderen WAS Komponenten unabhängig ob sie auf einem Rechner laufen oder in einer Serverumgebung auf mehreren Rechner verteilt installiert sind. Der WRB steuert auch die Kommunikation zwischen den Cartridges und den von Ihnen unterstützten Programmen. Dabei verwendet ein Cartridge einen oder mehrere sog. Webagents. Ein Webagent wird für einen Benutzer in der Datenbank angelegt. Mit ihm werden auch alle Pakete, mit deren Hilfe der Anwendungsprogrammierer den Weboutput erzeugen kann installiert. Dieses sind im Falle einer PL/SQL Webagentinstallation u.a. die htp und htf Pakete. Der Zugriff des WAS auf die Datenbank wird mit Hilfe eines sog. Database Access Discriptors (DAD) gesteuert. Der DAD beinhaltet die Zugriffsverwaltung auf die Datenbank, wie z.B. Zugriffsrechte. Diese beschriebenen Hauptkomponenten, insbesondere der WRB lassen sich zwar in weitere Unterkomponenten unterteilen, aber sie sind für das Verständnis des Web Application Servers im Ramen dieser Arbeit nicht erforderlich, so daß ich hier darauf verzichte die einzelnen Prozesse zu beschreiben.

Es sei allerdings darauf hingewiesen, daß der Web Application Server aus einer ganzen Reihe von Prozessen besteht, die untereinander in Beziehung stehen.

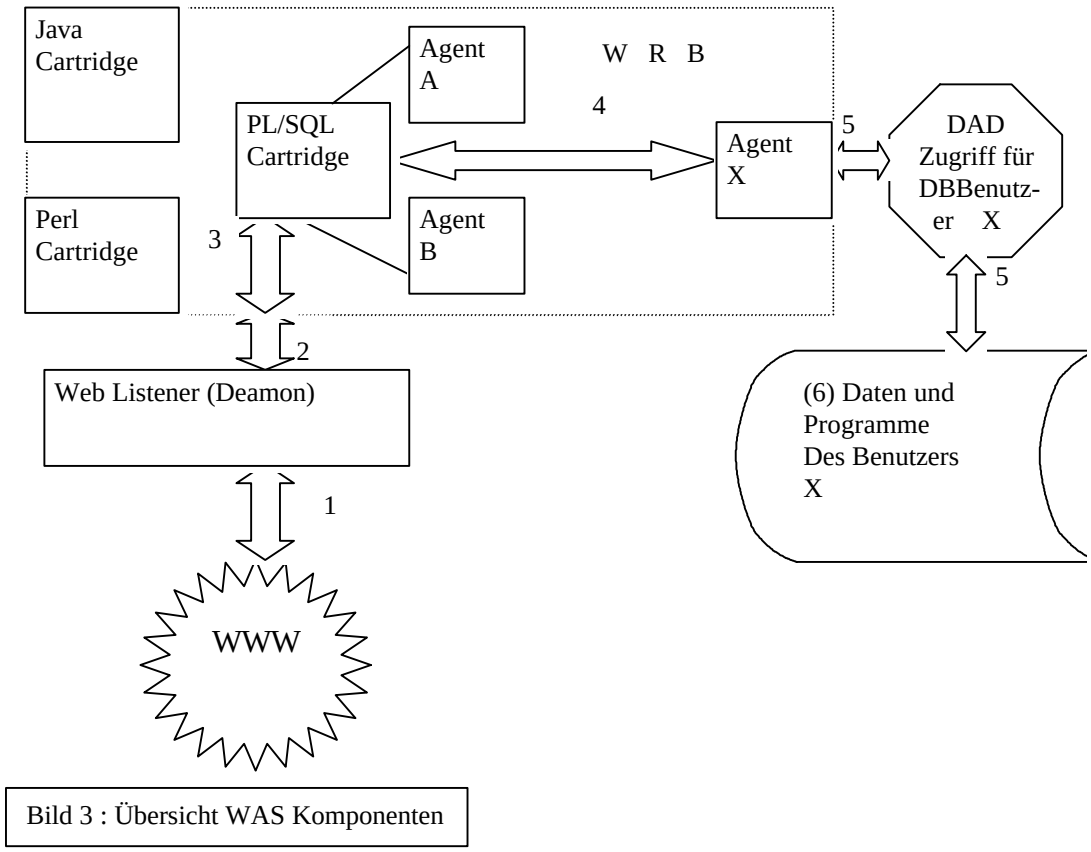


Bild 3 : Übersicht WAS Komponenten

Das obige Schema stellt alle im vorigen Abschnitt beschriebenen Komponenten im Aufbau und in ihren Beziehungen untereinander nochmals dar.

(1) Der Web Listener steht mit WWW in Verbindung und erhält eine Anfrage nach einer Seite bzw. Resource, da sich hinter einer angefragten URL eine dynamisch, von einem Programm zu erstellenden Internetseite handelt. Weiterhin erhält der Listener Informationen, die später im Programm als Parameter verarbeitet werden sollen.
(2) Der Web Listener übersetzt die angefragte URL und untersucht für welches Cartridge sie bestimmt ist. Hier nehmen ich einmal an das sie für das PL/SQL Cartridge bestimmt ist. Diese Informationen werden an das WRB System weitergegeben.
(3) Der WRB untersucht die Informationen weiter und bestimmt für welches PL/SQL Programm sie bestimmt ist, z.B. für eine bestimmte PL/SQL Prozedur, die in der Datenbank gespeichert ist.
(4) Das PL/SQL Cartridge ist mit mehreren Webagents verbunden und gibt Anfrage nach Resource dem richtigen Webagent weiter. Hierfür benutzt es den Prozedurnamen und die Art und Anzahl der übergebenen Parameter, da PL/SQL Prozeduren und Funktionen überladen sein können. Dieses geschieht mit Hilfe des DAD im nächsten Schritt.
(5) Über die DAD Daten erhält der Webagent Zugriff auf die Daten und Programme des Datenbankbenutzers, der die Ressourcen zur Verfügung stellt. Wird eine passende Prozedur gefunden ruft das Cartridge das entsprechende PL/SQL Programm mit den aus der Anfrage bestimmten Parametern auf.
(6) Das Programm generiert WWW Seite indem sie das Programm mit den übergebenen Parametern verarbeitet und das Ergebnis ausgibt.

Das Ergebnis wird durch den WRB zurück an den Listener und zum Schluß als WWW Seite vom Listener an den Anfragebrowser via http zurückgegeben.

2.2. Installation des Web Application Servers

Der Web Application Server kann auf einem einzelnen Rechner installiert werden, wobei der Web Listener, das WRB und die Cartridges zusammen auf diesem Einzelplatz installiert werden. Dieses ist die einfachste Installationsart und wird auch hier beschrieben. Der Web Application Server kann aber auch auf mehreren Rechnern als verteilte Installation aufgespielt werden. Es ist möglich, den Web Application Server auf einem anderen Rechner als den Web Listener zu installieren. Um einer weiter verteilte Installation zu bekommen, können verschiedene Komponenten des Web Application Servers auf mehrere Rechner Verteilt werden. Diese Installation in einer Multiserverumgebung wird auch als Multinodeinstallation bezeichnet und kann für die Geschwindigkeitssteigerung bzw. bessere Ausnutzung der Hardwareressourcen nützlich sein. Diese Multinodeinstallation ist aber auch die komplizierteste Installation des Web Application Servers. Während der Installation werden einige Eingaben für die Konfiguration des Web Application Servers benötigt. Dieses sind Fragen z.B. nach dem Oracle Home Verzeichnis, nach später zu verwendenden Paßwörtern u.ä. Die meisten dieser Installationsparameter haben einen Standardwert, der in den meisten Fällen übernommen werden kann. Später in diesem Abschnitt wird näher auf diese Installationsparameter eingegangen. Bevor mit der eigentlichen Installation des Web Application Servers begonnen werden sollte, müssen einige Vorbereitungen getroffen werden.

Bei der Installation unter dem Betriebssystem Windows NT der Fa. Microsoft sollte man über Systemadministrationsrechte verfügen, um den WAS überhaupt installieren zu können (Bei einer Unixinstallation erst recht). Außerdem gehe ich davon aus, daß eine Oracle 8 Datenbank auf dem Rechner installiert und für Benutzer eingerichtet ist. Bei der späteren Administration des Web Application Servers vor allem bei der Einrichtung der Web Application Server Programmpakete für einzelne Benutzer sollte man über DBA Rechte verfügen. Weiterhin sollte vor der Installation des WAS sichergestellt sein, daß die Oracle 8 Datenbank mit den net80 Netzwerkkomponenten installiert ist. Beim Installieren des WAS muß darauf geachtet werden daß die SQL*Net 80 Verbindungsoption gesetzt wird, da die Verbindung zwischen der Datenbank und dem WAS über diese SQL*Net Verbindung erfolgt. Außerdem sollte man die Datei sqlnet.ora im Oracle Homeverzeichnis auf den Eintrag TRACE_CLIENT_LEVEL = ON überprüft werden und gegebenenfalls hinzugefügt werden. Nun kann mit dem Setupprogramm auf der WAS Installations-CD der Installationsprozess gestartet werden. Während der Installation werden, wie schon erwähnt, mehrere Eingaben erforderlich. Das sind Fragen nach der Nationalsprache für die Unterstützung des Webservers sowie der Nachrichtentexte bzw. Meldungstexte des WAS – die Standardeinstellung hierbei ist Englisch. Ist auf dem Rechner keine Oracleinstallation vorhanden wird nach dem Firmennamen gefragt, für die diese WAS Installation lizenziert werden soll. Diese Frage stellt sich in diesem Beispiel nicht, da ja schon eine Oracle 8 Datenbank auf dem Rechner vorhanden ist.

Als nächstes wird nach dem Oracle Home Verzeichnis gefragt, dieses ist unter NT Standardmäßig das Verzeichnis C:\ORANT\. Nun wird nach dem Verzeichnis gefragt unter welchen die WAS Dateien installiert werden sollen, das Standardverzeichnis ist hier C:\ORANT\OWS\3.0\. Der letzte Verzeichnispfad, der benötigt wird, ist der wo die Konfigurationsdateien des WAS abgelegt werden sollen – hier ist das Defaultverzeichnis C:\ORANT\OWS\WEBSIDE30\. Als nächstes wird nach dem Hostname des Rechners gefragt, dieser kann unter NT in der Netzwerkumgebung nachgeschlagen werden bzw. vom Netzwerkadministrator erfragt werden. Bei einer Multinodeinstallation des WAS würden nun Fragen nach der Remote Host List, dem UDP Serviceport und dem shared key folgen. Diese Fragen stellen sich aber bei dieser Einzelplatzinstallation natürlich nicht. Als nächstes wird nach dem Port gefragt, unter dem später der Administrativlistener laufen soll. Standardwert ist der Port 8888 dieser kann aber verändert werden. Außerdem wird nach einem Benutzernamen und einem Paßwort gefragt, welche später für die Sicherheit beim Zugriff auf dem Administrativlistener verwendet wird. Zum Schluß wird nach dem Namen und den Port des Standardlisteners gefragt. Hier sind die Defaultwerte Port 80 und www. Jetzt, wenn alle Einstellungen vorgenommen worden sind, werden die restlichen Dateien installiert. Ist die Installation beendet sollte der Rechner neu gestartet werden, um die neuen Einstellungen wirksam werden zu lassen. Nach erfolgtem Neustart des Rechners kann mit der Administration und der Einrichtung des WAS für die Benutzer bzw. Anwendungsprogrammierer begonnen werden.

2.3. Administration der WAS Komponenten

Um den Web Application Server zu administrieren, müssen seine Prozesse bzw. Dienste gestartet werden. Es ist zwar möglich diese aus dem Windows NT Menü Dienste heraus zu starten bzw. zu stoppen, laut der Oracle Dokumentation sollten sie aber besser manuell gestartet bzw. gestoppt werden. Um aber trotzdem eine gewisse Vereinfachung bzw. Automatisierung beim Starten der Dienste zu erreichen , kann man folgendermaßen vorgehen. Es gibt drei Dienste die in der folgenden Reihenfolge gestartet werden müssen bzw. in umgekehrter Reihenfolge gestoppt werden müssen. Dies sind erstens der Dienst WRB, daß sind sämtliche web request broker Dienste bzw. alle Verwaltungsdienste des Web Application Servers zusammengefaßt, als zweites der admin Listener und zum Schluß der WWW Standardlistener. Hierzu wurde bei der Installation der Befehl owsctl angelegt, der über ein DOS Fenster die Dienste starten kann. Per Hand wird der Web Application Server gestartet, indem man ein DOS Fenster unter Windows NT öffnet und folgende Eingaben macht: owsctl start wrb <CR> owsctl start admin <CR> owsctl start www <CR>. Zum Stoppen der Dienste owsctl stop www <CR> owsctl stop admin <CR> owsctl stop wrb <CR>. Außer den start und stop Parameter kann man den owsctl Befehl mit dem Schlüsselwort status aufrufen, um zu überprüfen ob der Dienst gerade ausgeführt wird. Das Starten und Stoppen der Dienste sollte der Einfachheit halber in einem Batchfile geschrieben werden und ein start Batchfile kann bei Windows NT auch in den Autostart Ordner kopiert werden um die Dienste automatisch bei jedem Start des Rechners zu aktivieren.

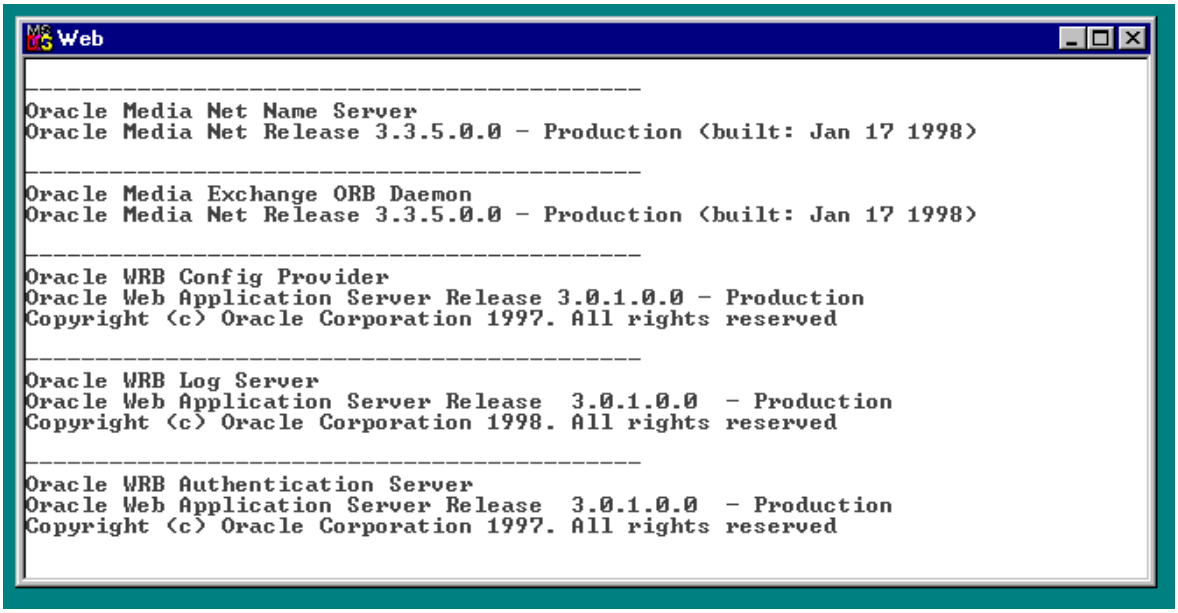


Bild 4: Hochfahren der WAS Dienste in einer DOS Box

Beim Hochfahren der Dienst wird das Starten der einzelnen Prozesse in der DOS Box angezeigt. Das obige Bild zeigt einige Ausgaben beim Starten des WRB. Nachdem so also die drei Dienste des Web Application Servers gestartet worden sind, werden als erstes die Benutzer der Datenbank für den Web Application Server eingerichtet. Erst dann können sie ihre Webanwendungen mit dem Web Application Server erstellen bzw. über seine Weblistener ins WWW stellen. In diesem Beispiel wird ein Benutzer für die Verwendung des PL/SQL Cartridges eingerichtet. Der Zugriff auf die gesamte Administration kann vom Administrativlistener des WAS vorgenommen werden und zwar von jedem Rechner mit einem WWW Browser, der ans Internet angeschlossen ist. Dieses ist von Oracle nur konsequent in der Konzeption des WAS (thin client bzw. reine serverseitige Verarbeitung). Um den admin aufzurufen wird die URL des Admindaemons angegeben. Diese setzt sich aus dem Hostname des WAS Rechners, der Portnummer des Administrativlisteners und der Internetdomäne des Rechners zusammen z.B.: http://ci2p3.gm.fh-koeln.de:7777.

Je nach Sicherheitseinstellung wird nach dem Benutzernamen und dem Paßwort für den Zugriff auf diese URL gefragt. Diese Einstellungen wurden vorher bei der Installation vorgenommen. Nun sollte die Begrüßungsseite des WAS Administrativlisteners erscheinen. Von hier aus kann man über mehrere Links z.B. zur Onlinedokumentation des WAS oder zur Oracle Internetseite kommen. Für die Einstellungen des WAS benutzt man den Link „Web Application Server Manager„.

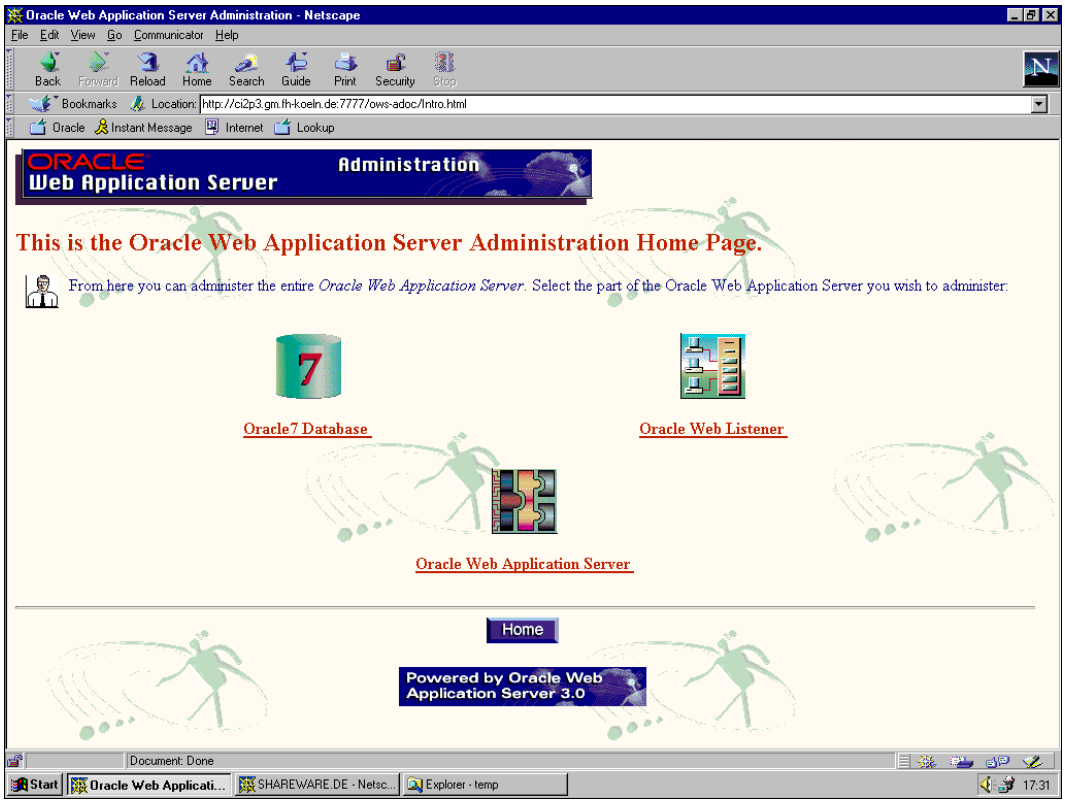


Bild5: Administrationsseite des WAS AdminListeners.

Von hier aus hat man Zugriff auf drei weitere Links und zwar für Einstellungen der Datenbank, der Weblistener und des Oracle Web Application Server (s.o.). Der letzte Link sollte nun aufgerufen werden. Um nun die Benutzer einzurichten, muß als erstes der DAD eingerichtet werden. Hierzu geht man über den Link „DAD Administration„ auf dieser Seite und klickt anschließend „Create new DAD„ an. Jetzt erscheint eine Liste, in der die Einstellungen des neuen DADs für den Datenbankzugriff vorgenommen werden können.

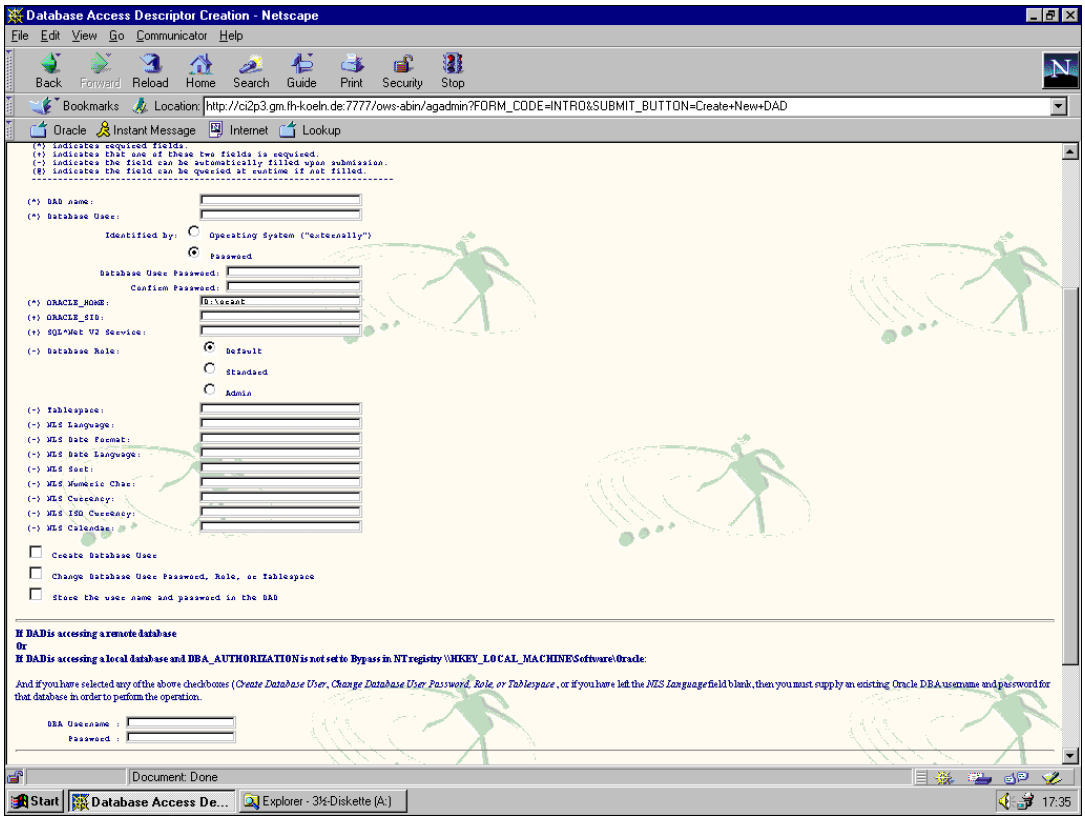


Bild6: Eingabemaske für die Erstellung eines neuen DAD

Alle Eingaben, die mit einem (*) Symbol gekennzeichnet sind müssen ausgefüllt werden, damit der neue DAD erstellt werden kann. Der Eintrag DAD name kann frei vergeben werden und identifiziert den neuen DAD. Unter dem Eintrag Database User wird der Name des Oracle Datenbankaccounts angegeben, für den der DAD erstellt werden soll. Der Eintrag „Database User Pasword“ ist das Paßwort des Benutzers bzw. des Datenbankaccounts, den man im vorherigen Eintrag vorgenommen hat.

Im nächsten Feld muß das Paßwort noch einmal bestätigt werden (Confirm pasword). Über den beiden Paßworteinträgen gibt es eine Schaltfläche „Identified by“. Hier ist „Pasword“ zu markieren. Als nächstes wird nach dem Oracle Home Verzeichnis gefragt. Dieses ist unter Windows NT standardmäßig C:\orant\. Das nächste Feld ist nur wichtig, wenn eine ältere Oracle Version als 8.0 benutzt wird und die Datenbank als auch der WAS auf demselben Rechner installiert sind. Ist dies der Fall wird hier die Datenbank SID eingetragen. Da hier die Oracle Version 8 benutzt wird kann man das Feld freilassen, muß aber das nächste Feld ausfüllen. Hier wird nach dem SQL*Net V2 Servicenamen gefragt. Diesen Datenbankname oder Datenbankalias findet man mit Hilfe des SQL*Net Easy Configuration Tools. Bevor der DAD erstellt werden kann sind ganz unten auf der Seite die Felder DBA Name und Paßwort auszufüllen. Dann kann das Formular durch anklicken des Submit Knopfes übertragen werden. Sind alle Einträge richtig, insbesondere die angegebenen Paßwörter erscheint kurze Zeit später eine Erfolgsmeldung und der DAD ist erstellt. Ist ein Eintrag nicht richtig ausgefüllt, erscheint eine Fehlermeldung und der Vorgang ist ggf. zu wiederholen. Auf der Seite mit dem Link „Create new DAD“ sollte jetzt ein weiterer Eintrag stehen, nämlich der Name des neu erstellten DAD. Wenn man auf diesen Link klickt, kann man den erstellten DAD auch nachträglich verändern. Gegebenenfalls können jetzt die weiteren DADs nach dem beschriebenen Muster für andere Benutzer erstellt werden. Ist der DAD erstellt muß ein Webagent für den Benutzer erstellt werden. In diesem Fall soll ein PL/SQL Webagent erstellt werden. Jeder Webagent benutzt um auf die Datenbank zuzugreifen einen DAD. Deshalb wurde zuerst der DAD erstellt.

Um nun für einen Benutzer einen Webagent zu erstellenm, geht man auf die WAS Konfigurationsseite zurück. Hier wird der Link „Cartridge Administration„ angeklickt. Weiter geht's mit dem Link „PL/SQL Cartridge„. Von hier aus kann jetzt ein neuer PL/SQL Webagent erstellt werden. Der PL/SQL Webagent beinhaltet Informationen für das PL/SQL Cartridge, um über den DAD auf die Datenbank zuzugreifen. Mit dem Erstellen des PL/SQL Webagenten werden die WAS Programmpakete für den Datenbankbenutzer installiert. Der Link „Create new PL/SQL Agent" führt zu einer Seite mit Eingabefeldern, in denen die Informationen für die Erstellung des Webagent angegeben werden.

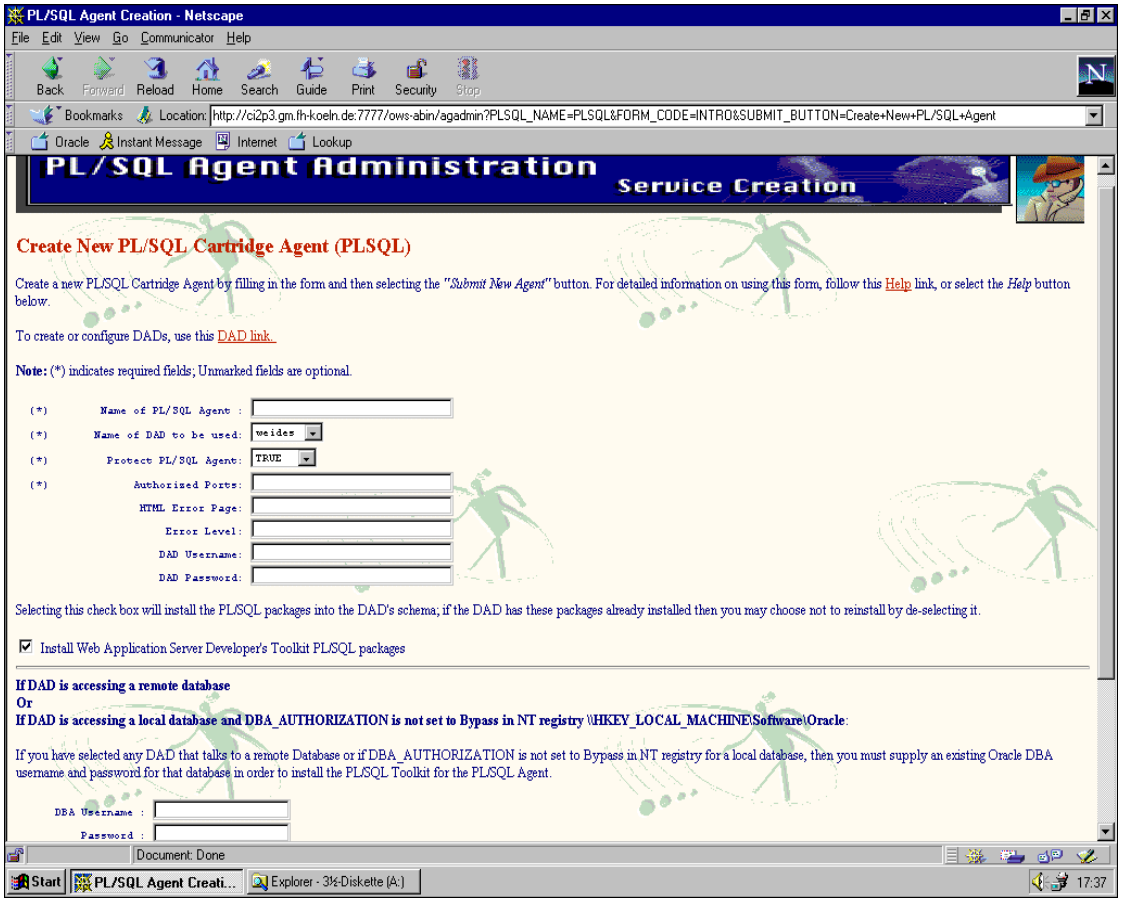


Bild7: Eingabemaske für die Erstellung eines WAS Webagents

Man vergibt einen Namen für den neuen Webagent und trägt den Namen des DAD ein, mit dem der Agent Verbindung zur Datenbank aufnehmen soll. Um die Programmpakete zu installieren, markiert man die Checkbox „Install WAS developer's toolkit PL/SQL packages". Auch hier gibt man am Ende der Seite einen DBA Benutzernamen und das Paßwort an. Nun überträgt man wieder die Informationen an den Weblistener, indem man den Submitknopf anklickt. Sind alle Einträge richtig werden die Programmpakete in dem im angegebenen DAD festgelegten Benutzeraccount der Datenbank kopiert. Dieser Vorgang dauert einige Minuten. Der Fortschritt der Paketinstallation wird auf dem Bildschirm angezeigt. Ist der neue Webagent eingerichtet erscheint eine Erfolgsmeldung. Auch neu eingerichtete Webagents kann man jederzeit ändern, ähnlich wie bei dem DAD wird der neue Webagentname als Link angezeigt. Um die Einrichtung des neuen Webagents und DAD wirksam werden zu lassen, muß man die WAS Dienste stoppen und wieder neu starten. Jetzt kann der Datenbankbenutzer PL/SQL Funktionen und Prozeduren schreiben und mit Hilfe der vom Web Application Server installierten Pakete dynamische HTML Seiten erzeugen. Die wichtigsten Web Application Server Pakete sollen im Kapitel 3 Anwendungsentwicklung mit den Web Application Server Tools vorgestellt werden.

2.3.1. Administration des Weblisteners

Der Weblistener, der sich bei dem Web Application Server Paket befindet, läßt sich ebenfalls konfigurieren. Die Einstellungen lassen sich, wie die des Web Application Servers, über den WAS Administrationslistener vornehmen.

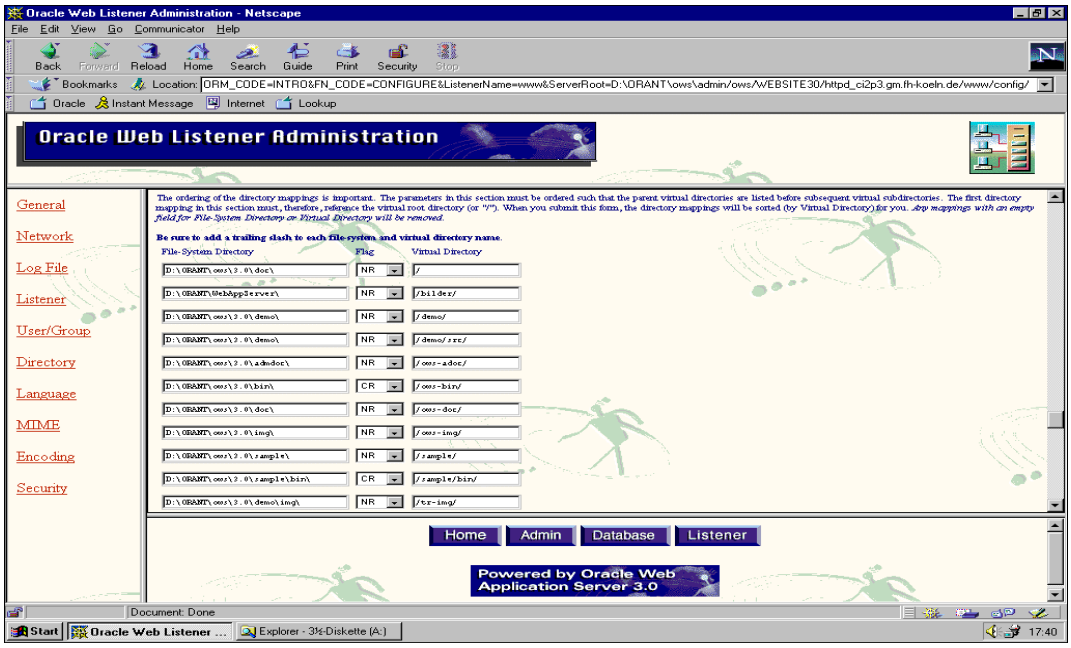


Bild8: Administrationsseite des Weblisteners

Um nun Einstellungen an den Listenern durchzuführen, folgt man dem Link auf der WAS Administrationsseite „Oracle Web Listener“, der sich auf der rechten Bildschirmseite befindet. Jetzt erscheint eine Seite mit dem Titel „Oracle Web Listener Administration„. Von dieser Seite aus kann man weitere Listenerprozesse erzeugen, die auf einem anderen Port dieses Rechners laufen können. Von hier aus ist es ebenfalls möglich externe Listener, also nicht Oracle Listener, wie z.B. von Microsoft oder Netscape, für den Gebrauch mit dem WAS zu registrieren. Nach dieser Registrierung können diese Listener ebenfalls von der Web Listener Administration Seite aus konfiguriert werden.

Der Netscape Server und der Microsoft IIS für WAS 3.0 sind vorgegeben, man kann aber auch Listener von anderen Firmen suchen lassen. Bei der Installation des Web Application Servers werden der Standart WWW Listener und der Administrationslistener standardmäßig erstellt. Ganz unten auf dieser Seite werden alle für den Web Application Server registrierten Listener tabellarisch angezeigt. Dieses sind also standardmäßig die beiden Listener, die bei der Installation des Web Application Servers erstellt wurden. Sie werden durch ihren Namen, also WWW oder admin identifiziert. Jeder, der hier aufgeführten Weblistener kann von hier aus gestartet (start), gestoppt (stop) und gelöscht (delete) werden. Außerdem wird angezeigt, ob der Weblistener gerade aktiv ist (blinkendes running!). Außerdem wird für jeden Listener angezeigt auf welchen Port er arbeitet und an welche IP Adresse er listen darf (Standardmäßig jede „any IP„). Diese letztere Einstellung wird für Sicherheitszwecke benutzt, mehr dazu später in diesem Kapitel. Um einen Listener zu konfigurieren, klickt man auf den Link „configure“ in der Zeile, die den Namen des gewünschten Listeners enthält. Bei der Konfigurierung der Listener sollte man auf zwei Dinge achten. Werden Einstellungen des Listeners verändert, dieses erfolgt wie bei allen Einstellungen über den Administrativlistener durch ein Submit, muß der Listener neu gestartet werden, um die Änderungen wirksam werden zu lassen. Weiterhin sollten Änderungen der Listenereinstellungen, besonders die des Administrativlisteners, mit Vorsicht vorgenommen werden. Falsche Einstellungen z.B. fehlerhaft Pfadeingaben führen in der Regel dazu, daß sich der betroffene Listener nicht mehr starten läßt. Im Falle des Administrativlisteners führt das dazu, das man die fehlerhaften Einstellungen nur schwer rückgängig machen kann (Ändern der Einstellungen des CFG-files des Listeners s.u.).

Auf der Konfigurationsseite des Listeners können alle Parameter verändert werden, außer die Sicherheitseinstellungen, für die es eine weitere Einstellungsseite mit dem Namen „Oracle Web Listener Administration – Server Advanced Configuration„ gibt – dazu später. Die Listener Konfigurationsseite ist in mehrere Abschnitte unterteilt. Die „General Information“ - hier wird kurz auf die Funktion der Eingabeforms dieser Seite eingegangen. „Network Parameter“ – hier können die Kommunikationsverfahren des Listeners, wie Adressen und Portkombinationen, eingestellt werden. Man kann wählen, ob über Proxy gelistet werden soll, die Portnummer des Listeners kann zwischen 1 und 65535 eingestellt werden, es können Sicherheitseinstellungen wie SSL oder PCT Port vorgenommen werden. Der Hostname des Rechners auf dem der Listener arbeitet kann verändert werden, sowie das Standart URL Verzeichnis (Default : /). Außerdem kann der Pfad des Log Files des Listeners verändert werden. Weiterhin können Sicherheitseinstellungen für die Clients vorgenommen werden. Es ist möglich nur an bestimmte , zertifizierten Clients zu listen. Weiterhin kann die Zahl der maximal gleichzeitig zu bearbeitenden Anfragen gesetzt werden (Default: 338). Durch die Einstellung „redirect Server“ können bei höheren Anfragevolumen Anfragen, die diesen Maximalwert überschreiten an andere Listener weitergegeben werden. Es kann weiterhin eingestellt werden, ob der Listener die in den Anfragen enthaltenen Hostnamen der Rechner über den „Domain Name Service“ auflösen soll, die wird im Einstellungsteil „DNS Resolution“ eingestellt. Die nächste Einstellung betrifft das PID File des Listeners. Diese Datei identifiziert den aktuellen Listener Prozess. Dieses File befindet sich für den WWW Standardlistener im Defaultverzeichnis:

oraclehome\ows\admin\ows\WEBSIDE30\httpd_hostname\www\config\
und ist unter den Namen svlistenername.pid (im Falle des WWW Listeners also svwww.pid) abgelegt. Sollen Änderungen in dieser Einstellungsdatei vorgenommen werden muß der entsprechende Listener heruntergefahren werden, da er nach einer Änderung der Datei nicht ohne weiteres heruntergefahren werden kann. Die nächste Einstellung der OWAS Listener Administration betrifft das Log File. Hier werden alle Informationen bzw. Fehlermeldungen beim herauf- bzw. herunterfahren der Listener angezeigt und kann deshalb für die Untersuchung von Fehlern beim Betrieb des WAS herangezogen werden. Das Standardverzeichnis des Files ist: oraclehome\ows\admin\ows\WEBSIDE30\httpd_hostname\www\log\ und der Filename ist svlistenername.err (bzw. Beim WWW-Listener: svwww.err). Für dieses File kann eingestellt werden wann die Informationen überschrieben werden sollen: nie, monatlich, wöchentlich oder täglich. Der Zeitstempel der Datei können auf GMT oder lokale Zeit gesetzt werden. In einem weiteren Abschnitt kann eingestellt werden welche Einträge in das Logfile vorgenommen werden sollen. Die Einstellungen im nächsten Abschnitt beziehen sich auf das Konfigurationsfile des Listeners. Die Konfigurationsdatei enthält alle Einstellungen des Listeners. Die Datei ist unter folgendem Verzeichnis standardmäßig abgelegt:
oraclehome\ows\admin\ows\WEBSIDE30\httpd_hostname\www\config\
Der Name der Datei ist svlistenername.cfg (bzw. Beim WWW Listener svwww.cfg). Falls durch fehlerhafte Einstellungen ein Listener nicht mehr gestartet werden kann, ist es möglich, durch setzen der Werte dieser Datei fehlerhafte Einstellungen rückgängig zu machen.

Es ist empfehlenswert diese Datei zu sichern, um bei Problemen mit dem Listener die Defaultwerte bei Hand zu haben. Die nächste Einstellung bezieht sich auf das Initial File. Wird bei Browseranfragen kein Dateinamen angegeben wird diese Seite zurückgegeben. Die Einstellung ist standardmäßig index.html. Die nächsten Abschnitte beinhalten Einstellungen nach dem User Directory (nur für Unix Installationen), dem Default Mime Typ, für den Fall das ein nachgefragter Mime Typ nicht unterstützt wird, dem Stadardzeichensatz und der bevorzugten Sprache mit der z.B. Dateiextensions angelegt werden sollen. Die nächste Einstellungen legt die Dateierweiterung für Bilddateien fest, die für sog. Image Maps benutzt werden sollen. Solch ein Image Map ist eine Grafik, die in eine Webside eingebunden werden kann und ein oder mehrere Submitfunktionen (funktioniert wie der normale Submitknopf, allerdings in einer Grafik bzw. Bild eingebunden) enthält. Die Standarderweiterung ist .map. Die nächsten Einstellungen beziehen sich auf diverse Timeout Einstellungen und auf Einstellungen für den Unix Betrieb. Der nächste wichtige Abschnitt in den Listenereinstellungen ist der Bereich der Directory Mappings. Die wichtigste Funktion eines Weblisteners ist die Übersetzung von physikalischen Dateipfaden, in denen sich html-Files befinden in das virtuelle Dateisystem des Listeners, um so die angefragten URLs identifizieren zu können. Die Einstellungen sind hier in Listenform gehalten. Die erste Spalte beschreibt den physikalischen Pfad der Webresourcen. Möchte man z.B. ein Bild (.gif , .bmp oder ähnliches) in eine Webseite integrieren, kann man das mit dem <IMG> tag machen. Dieser tag benötigt aber einen virtuellen Pfad, um den physikalischen Standort der Datei auszumachen.

Hier könnte man in den Einstellungen des Directory mappings z.B. einen Pfad festlegen, in dem solche Bilddateien abgelegt werden sollen z.B. d:\ORANT\bilder\. Die nächste Spalte fragt, ob der angegebene Pfad CGI Scripts enthält, und in welcher Weise der Pfad gemappt werden soll. Da unser Beispielspfad nur Bilddateien enthält bekommt diese Einstellung den Wert NR für keine Scriptdateien und sog. Rekursiv-mapping. In der dritten Spalte steht nun der Virtuelle Pfad z.B./bilder/ mit dessen Hilfe die Bilder später mit dem <IMG> Tag eingebunden werden können. Die nächsten Einstellungen des Listeners beziehen sich wieder auf Zeichensätze und Mime Typen Einstellungen. Die letzten Einstellungen beziehen sich auf die Sicherheitseinstellungen des Listeners. Um auf diese Einstellungen zuzugreifen klickt man auf den Link „Security“ , ganz unten im linken Frame der Oracle Web Listener Administrationsseite. Den Zugriff auf die mit dem Oracle Web Listener gelisteten Seiten kann man mit Hilfe von vier Eingabeforms einstellen. Zum einen gibt es die Authentications Forms „Basic Authentication“ und „Digest Authentication“. Diese Formulare bestehen aus Listen, in denen man Benutzer und Passwörter einrichten kann. Die Benutzer können anschließend zu Gruppen zusammengefaßt werden und verschiedene Gruppen wiederum zu Obergruppen zusammengefaßt werden, den sog. „realms“. Den Benutzern, Gruppen und Obergruppen kann man nun Zugriff auf bestimmte Dateien und Verzeichnissen mit Hilfe ihres Benutzernamens und Paßwortes gestatten. Der Unterschied zwischen den „Basic“ und „Digest“ Authenticationeinstellungen besteht darin, das die Basic-Authentication den Benutzernamen und das Paßwort unverschlüsselt über das Netz sendet, also eine schwächeren Schutz bietet, während die Digest Einstellung die Paßwörter mit einem Kryptografischen Checksummencode versendet und daher mehr Sicherheit bedeutet.

Weiterhin gibt es die Möglichkeit, Webseiten und Verzeichnisse nur für Rechner mit einer bestimmten IP-Adresse zugänglich zu machen. Diese kann auch zusätzlich zu einem bereits bestehenden Benutzer/Paßwortschutzes eingerichtet werden. Als letztes ist es noch möglich Zugriff auf Webseiten und Verzeichnisse mit Hilfe von Domain-Namen zu ermöglichen. Der Schutz einer bestimmten Webseite erfolgt nun im Abschnitt „Protection“. Dieses Eingabeform ist ebenfalls in Form einer Liste gehalten. Hier wird als erstes der Virtuelle Pfad der zu Schützenden Webseite eingegeben, dann der Zugriffsmodus (Lesen, Schreiben, Löschen bzw. in http das Erlauben der GET,HEAD oder POST Methoden auf diese Seite zum Lesen, PUT zum Schreiben bzw. DELETE zum Löschen). Als nächstes wird kann die Art der Autentication, also entweder Basic oder Digest und besondere Verschlüsselungsarten, entweder über den SSL (secure socket layer) oder den PCT eingestellte werden. Im nächsten Feld kann mit den Symbolen & bzw. | eingestellt werden ob nur eine Benutzername/Paßwort Identifikation oder zusätzlich eine IP bzw. Domain Identifikation durchgeführt werden soll. Das Symbol & bedeutet, das sowohl der Benutzername/Paßwort als auch die IP Adresse bzw. Domain für den Zugriff herangezogen werden, das Symbol | bedeutet das entweder der Benutzername/Paßwort oder die IP bzw. Domain zum Zugriff berechtigt. Die nächste Spalte IP/Domain kann eingestellt werden ob der Zugriff über eine IP Adresse oder Domain gesteuert werden soll, entsprechende Gruppen für die Identifikation der IP Adresse bzw. der Domain müssen vorher s.o. festgelegt worden sein. Dieses Feld kann auch freigelassen werden, wenn der Zugriff nur über einen Benutzeraccount gestattet werden soll.

In die letzte Spalte können nun die vorher festgelegten Gruppennamen für die IP-Adressen bzw. Domain eingetragen werden, für die der Zugriff erlaubt wird. Im letzten Form auf dieser Seite können Einstellungen für die Verschlüsselungsart vorgenommen werden. Eine im Web häufig benutzte Verschlüsselungsart ist das Verschicken und Empfangen von Sicherheitsrelevanten Daten über einen eigenen Port für den sog. Secure Sockets Layer oder SSL. Hierzu muß allerdings ein eigener Port für den SSL eingerichtet werden. Diese Einstellungen können auf der „Advanced Configuration“ Seite s.o. vorgenommen werden. Eine Webanwendung durch eine Benutzer/Paßwort Sicherung zu sperren ist aber praktisch nicht ganz so einfach, wie es in der Online Dokumentation beschrieben ist. Angenommen man hat eine PL/SQL Prozedur mit Namen beispiel.sql geschrieben, die durch ein Paßwort geschützt werden soll und die Anwendung wird durch den Webagent agentxyz verwaltet. Um diese Resource vor unbefugten Zugriff zu sichern, ruft man als erstes die Security Seite der Web Listener Administration auf. Hier wird dann, wie oben beschrieben, der/die Usernamen und das/die Paßwort(e) unter dem Abschnitt „Basic Users“ eingetragen. Die Benutzer werden dann unter „Basic Groups“ zu Gruppen zusammengefaßt z.B. unter dem Gruppennamen bseipielgroup. Unter „Basic Realms“ kann diese Gruppe z.B. der Obergruppe Anwender_1 zusammengefaßt werden. Unten auf dieser Seite befindet sich dann der Abschnitt „Protection“. Hier wird dann der Virtuelle Pfad der Anwendung, in diesem Fall /agentxyz/plsql/beispiel angegeben, um diese eine Anwendung zu schützen. Es können hier auch nur Pfade angegeben werden, wie z.B. /agentxyz/plsql/ , dann würden alle Ressourcen unter diesem Pfad geschützt. In der nächsten Spalte wird die Zugriffsart eingestellt, z.B. RWD (Read,Write,Delete).

In der nächsten Spalte wird die Verschlüsselungsart eingestellt, in diesem Fall „Basic“. In der folgenden Spalte wird die Obergruppe „realm“ angegeben, für die der Zugriff gestattet werden soll, in diesem Fall Anwender_1.

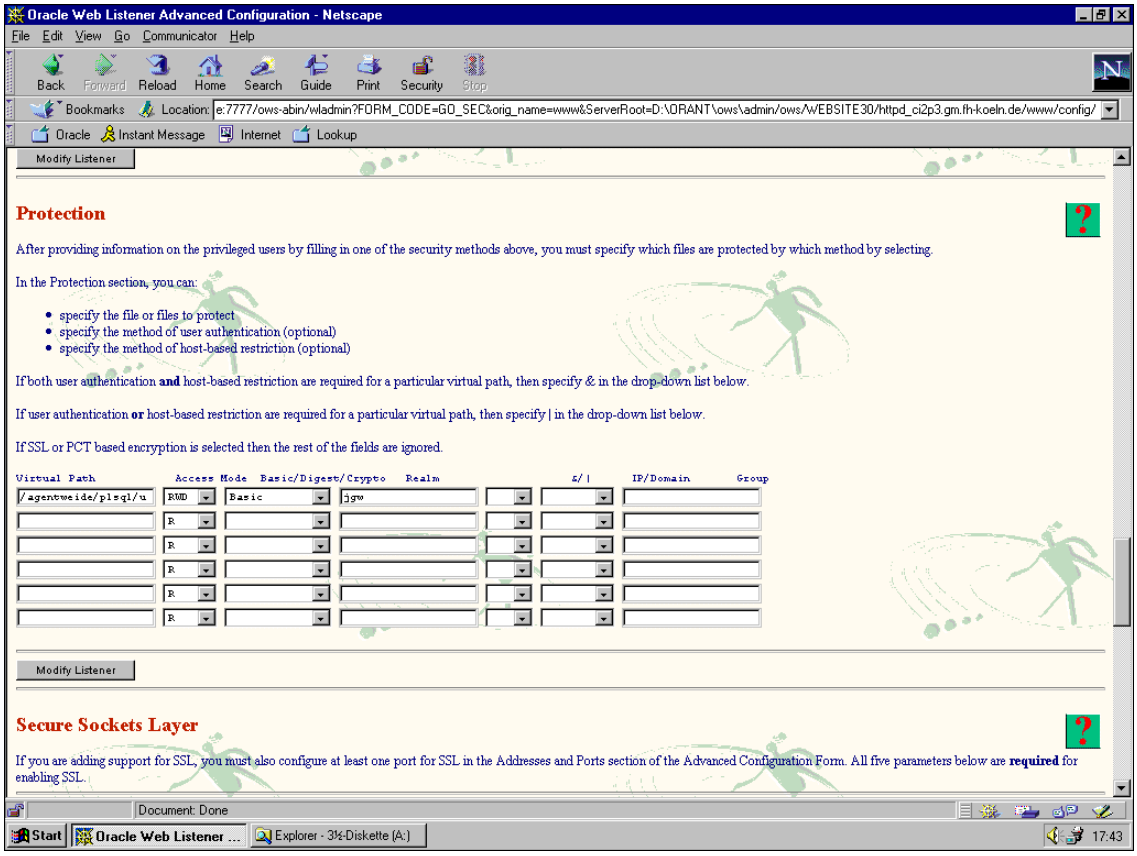


Bild9: Eingabemaske für das Schützen einer Anwendung oder virtuellen Pfads

Die nächsten Spalten können freigelassen werden, da in diesem Fall der Zugriff über IP-Adresse oder Domainname nicht gestattet werden soll. Über die Schaltfläche „Modify Listener“ werden die neuen Listenereinstellungen gespeichert und durch einen Neustart des WWW Listeners aktiviert. Mit diesen neuen Einstellungen ist die Resource „beispiel“ aber leider noch nicht geschützt, wie man es nach der WAS Dokumentation glauben kann.

Es gibt bei der WAS Administration noch zwei weitere Security-Forms, die für solch eine Paßwortabfrage ausgefüllt werden müssen, bevor sie auch wirklich funktioniert. Zum einen gibt es die Administrationsseite Web Application Server Authentication. (.../ows-abin/atadmin). Man erreicht diese Website indem man von der Administrations Home Page über den Link „Oracle Web Application Server“ auf den Eintrag „Authorization Server“ oben Links auf der Seite klickt. Hier erscheint wieder eine Liste von Links, die schon von der Security-Seite des Listeners bekannt sind, nämlich „Basic“ , „Digest“ , „IP-Based“ , „Domqain-based“ - sie haben dieselbe Bedeutung, wie auf der Securityseite des Listeners. Klickt man nun „Basic“ an, erscheint ein Eingabeform, auf dem, wie vorher ein Benutzername, Paßwort, eine Gruppe und eine Obergruppe angegeben werden muß. Sinnvoller Weise verwendet man hier dieselben Einträge, wie bei den Listener-Securityeinstellungen. Mit dem Submitknopf „Modify“ schließt man die Einstellungen ab. Als letzte Einstellungen sollte man sich nun die WRB Administration ansehen. Zu der WRB Administrationsseite führt leider kein Link direkt, deshalb gibt man die folgende URL direkt ein:

http://Hostname:AdminPortnr/ows-abin/wrbadmin/

Auf dieser Seite sind die wichtigsten Parameter für den Web Request Brooker des Web Application Server Systems aufgelistet. Hier findet man auch ein Eingabeform mit dem Titel : „Protecting Applications“. In die erste Spalte gibt man den virtuellen Pfad an, in diesem Beispiel also /agentxyz/plsql/beispiel. Das Verschlüsselungsschema, hier also Basic und den Obergruppennamen (Realm).

Außerdem kann man die Resource zusätzlich durch die Domain oder die IP Adresse des anfragenden Rechners schützen lassen. Mit dem Submit „Modify WRB Configuration“ werden die neuen Einstellungen geschrieben. Benutzernamen und sein Paßwort identifizieren, bevor er Zugriff erhält. Anschließend stoppt man die WAS Dienste und startet sie neu. Wird jetzt das Programm .../agentxyz/plsql/beispiel angefragt muß sich der Benutzer durch seinen Benutzernamen/Paßwort identifizieren, um die Webseite aufzurufen.

3.0 Anwendungsentwicklung mit den WAS Tools

Das Erstellen von PL/SQL Programmen erfolgt wie gewohnt. Man kann z.B. mit einem einfachen Texteditor den Quellcode schreiben und unter dem Oracle Servermanager mit dem @ Befehl den Code compilieren und in der Datenbank speichern.

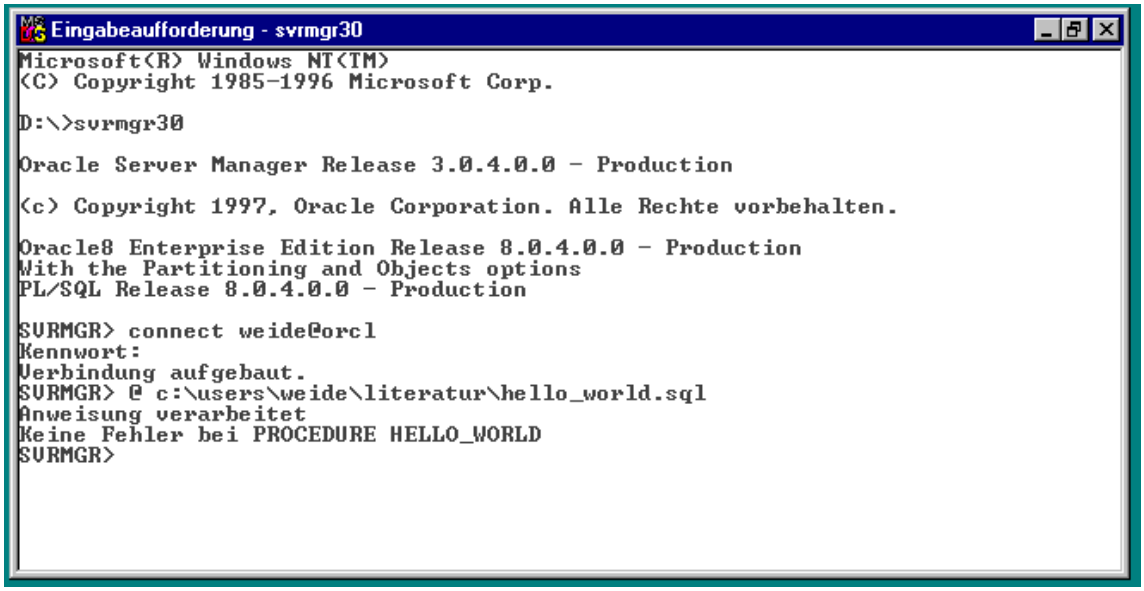


Bild10: Compilieren eines PL/SQL Programms mit dem svmgr30

Um ein PL/SQL-Programm, welches die WAS Werkzeuge benutzt, auszuführen öffnet man einen Internetbrowser und gibt die URL nach folgenden Muster ein:

http://hostname:Portnummer/agentname/plsql/programmname

bzw. adressiert die Links nach diesem Muster. Im PL/SQL Cartridge sind mehrere sog. Pakete oder Libraries enthalten, die bei der Installation des Webagents für den Anwendungsentwickler eingerichtet werden. Diese Pakete definieren Datentypen, Funktionen und Prozeduren, die vom Anwendungsentwickler für Webanwendungen verwendet werden können. Sie ermöglichen es dynamische HTML Seiten zu erzeugen, die Daten von der Datenbank enthalten. Im einzelnen gibt es folgende Pakete: Das htp und htf Paket für die Ausgabe, das cookie Paket für das Verwenden von Cookies, das image Paket für das Einbinden und Verwalten von Bildern in Webseiten, das opt lock Packte um das Problem des sog. Lost Update, welches bei der Veränderung derselben Daten durch mehrere Anwender gleichzeitig in der Datenbank auftreten kann, das Pattern Paket für das Auffinden von Textmustern, das sec Paket für die Entwicklung eigener Sicherheitsabfragen, das text Paket für das Verwalten großer Textmengen und das Util Paket. Auf die wichtigsten Pakete, wie das htp/htf Paket, die Funktionsweise des owa_opt lock Paket und das cookie Paket werden in den nächsten Abschnitten näher vorgestellt.

3.1. Die htp und htf Pakete

Die wichtigsten Pakete für den HTML Output ist das htp und das htf Paket. Sie werden wie alle anderen WAS Pakete bei der Erstellung des Webagent in den Benutzeraccount der Datenbank installiert. Mit den in den Paketen enthaltenen Prozeduren und Funktionen wird die Ausgabe in HTML Format vorbereitet. Die Ausgabe des htp Pakets werden in einen Puffer abgelegt. Dieses geschieht nach Abarbeitung des Programmcodes. Anschließend wird der Puffer gelehrt, das HTML erzeugt und über den Weblistener an den Anfragebrowser geschickt. Das htf Paket hat ähnliche Funktionen wie das htp Paket, die Ausgabe wird aber anders als beim htp Paket nicht in einen Puffer abgelegt, sondern in Varchar2 Format zurückgegeben. Die Programme im htf Paket sind daher Funktionen. Durch die unterschiedlichen Ausgaben kann man HTML Fragmente verbinden oder auf andere Weise manipulieren, bevor die fertige HTML Seite an den Browser zurückgeschickt wird. Ein einfaches hello_world Programm soll als Beispiel die Verwendung des htp Pakets zeigen. Außerdem kann man mit solch einem kleinen Programm feststellen, ob des DAD und der Agent richtig angelegt wurden. Hierzu tippt man z.B. in einen Texteditor folgende Zeilen ein:

```
CREATE OR REPLACE PROCEDURE hello_world
IS
BEGIN
htp.htmlOpen;                    /*Öffnet einen html Ausgabe Block*/
htp.headOpen;                    /*Öffnet den Kopftitelblock*/
htp.title('Hello World by Juri');  /*Legt den Überschriftstext fest*/
htp.headClose;                   /*Schließt den Überschriftskopf*/
htp.p('Hello World by J. Weide, the date is: '||SYSDATE);/*Erzeugt die Ausgabe des Datums*/
htp.htmlClose;                   /*Schließt den html Ausgabe Block*/
END;
/
show errors                      /*zeigt Compilierungsfehler an*/
```

Das Ganze speichert man ab z.B. unter C:\Programme\hello_world.sql. Nun kann man den Oracle Servermanager aufrufen, sich mit dem Benutzeraccount mit der Datenbank verbinden und diese Prozedur mit der Zeile:

@ C:\Programme\hello_world.sql

compilieren und in der Datenbank abspeichern lassen. Um das Programm nun ablaufen zu lassen, öffnet man seinen Internetbrowser und gibt die URL nach folgendem Muster ein:

http://Servername:Portnr/agentname/plsql/hello_world

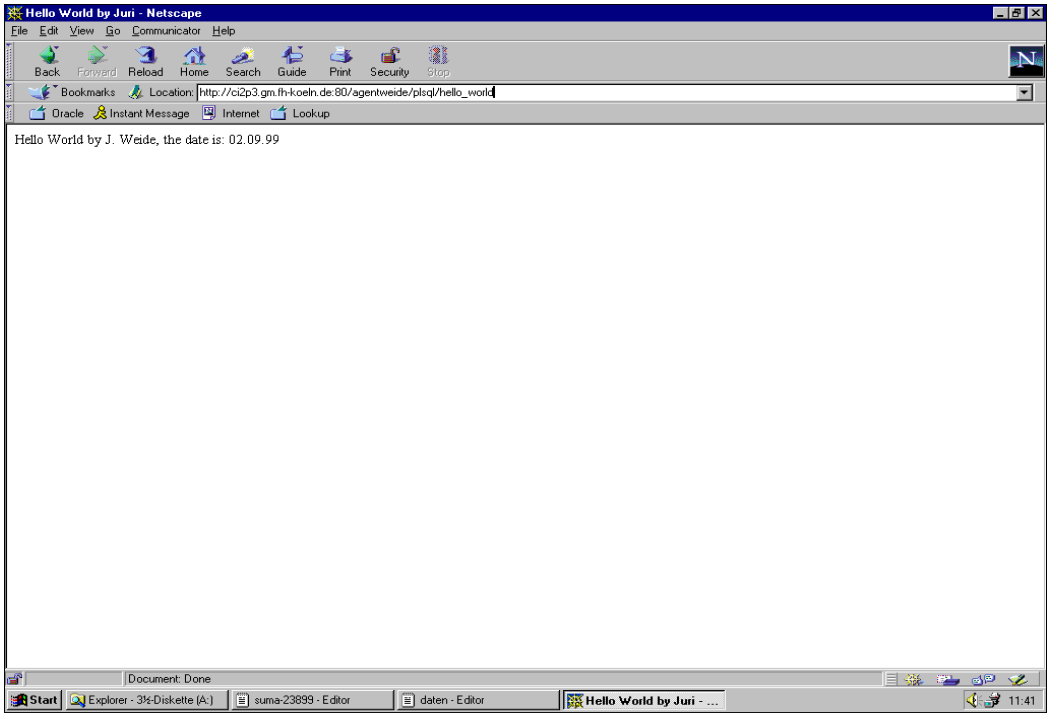


Bild11: Ausgabe des hello_world Programms

Nun sollte der Browser den Text, wie im obigen Abbild, anzeigen.

Eine wichtige Prozedur des htp Pakets ist die htp.p() Prozedur. Mit ihr kann man alle HTML Textformatierungen durchführen. Zwischen den Klammern bzw. als Argument der Prozedur kann man alle HTML tags verwenden z.B.: htp.p('<cite>Die ist ein Text</cite>');. Weiterhin beinhaltet das htp Paket Prozeduren für die Erstellung von Links und HTML Forms für die Erstellung von Eingabemasken. Für Links wird die htp.anchor2() Prozedur benutzt. Sie erzeugt eine Hyperreference wie in HTML die <A> bzw. </A> tags. Diese Prozedur verlangt allerdings immer die Angabe einer Ziel URL, so das ein Verweis auf derselben Seite mit ihr nicht möglich ist. Hier kann man sich aber mit der htp.p Prozedur behelfen, indem man die HTML tags mit ihr verwendet. Ein weiterer wichtiger Teil des htp Pakets sind Prozeduren und Funktionen für die Erstellung von Eingabemasken bzw. HTML-Forms Hier ein kurzes Programmfragment, welches die Verwendung von Forms verdeutlichen soll. Zum besseren Überblick sind hier Zeilennummern verwendet worden:

```
   ...  ...  ...  ...  ...  ...
1        htp.formOpen('Prozedurname_xyz');
2        htp.p('<center> Bitte wählen Sie aus:</center>');
3        htp.tableOpen('1','center'); --Formatiert eine Ausgabetabelle mit einer
4        htp.tableRowOpen('left','top'); --Spalte
5        htp.tableData(htf.formRadio('name','value1')||htf.bold('Datei hinzufügen')||htf.br||
6        htf.formRadio('name','value2')||htf.bold('Datei löschen')||htf.br||
7        htf.formRadio('name','value3')||htf.bold('Datei updaten')||htf.br||
8        htf.formRadio('name','value4','checked')||htf.bold('Zurück')||htf.br);
9        htp.tableRowClose;
10       htp.tableClose;
11       htp.formSubmit(NULL,'Submitknopf');
12       htp.p('</center>');
13       htp.formClose;
  ...  ...  ...  ...  ...  ...  ...
```

In Zeile 1 wird die Form, also die Eingabemaske geöffnet. Die Benutzereingaben werden, wenn der Benutzer später die Daten mit dem Submit an den Listener zurücksendet an eine Prozedur mit dem Namen Prozedurname_xyz geschickt und dort verarbeitet. Zeile 2 bewirkt eine einfache Informationstextausgabe. Zeile 3 erstellt eine Ausgabetabelle, eine Textformatierung in Form einer Tabelle mit einer Spalte, die auf der Seite zentriert erscheinen soll. Zeile 4 öffnet die erste Zeile der Ausgabetabelle und legt den Text linksbündig an. Nun erfolgt in den Zeilen 5 bis 8 die Programmierung eines sog. Radiomenüs, in dem der Benutzer exakt einen Menüpunkt auswählen muß. Die Bezeichnung name steht für den name des name/value Paars, welches später beim Submit versendet wird. Er ist natürlich in allen Menüpunkten derselbe, da exakt ein name/value Paar versendet werden soll. Die Ausprägung also das „value„ des name Parameters ist hier, je nach Auswahl value1 bis value4. Wobei das value4 standardmäßig als Vorauswahl ausgewählt wird, dieses geschieht mit dem Attribut checked. In Zeile 11 wird der Submitknopf programmiert. Das Menü sollte wie im unteren Abbild angezeigt werden.

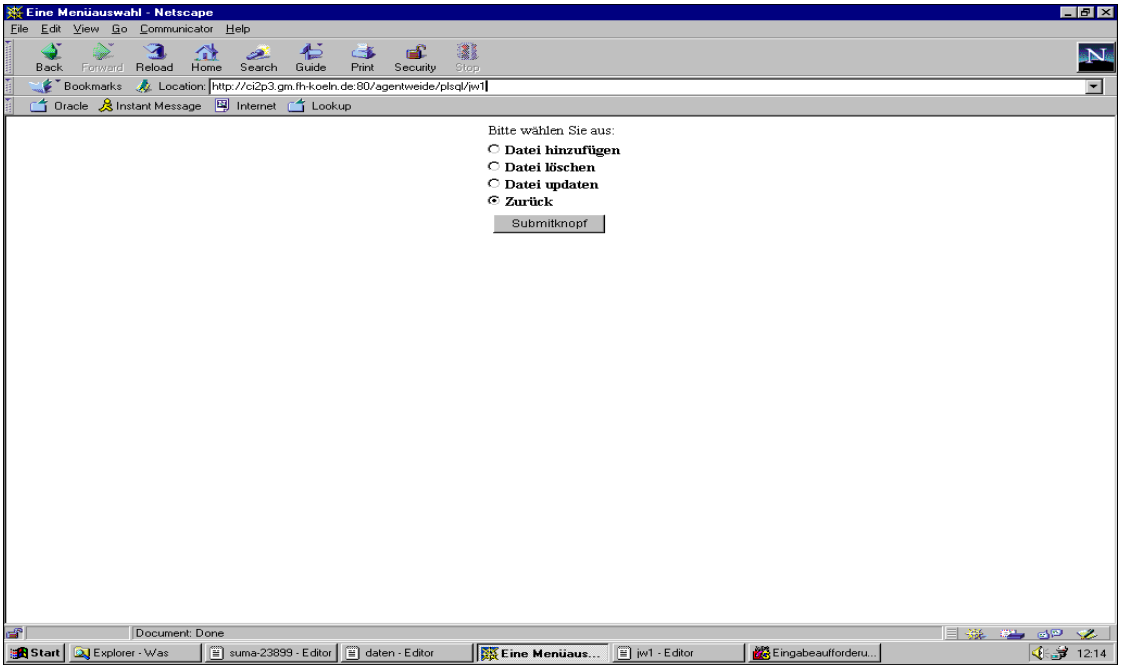


Bild12: Ausgabe eines Menüs

Das htp bzw. htf Paket besitzt noch eine ganze Reihe weitere Tools, die falls sie im Beispiel benutzt werden in der Programmdokumentation erklärt werden.

3.2. Das owa Optimistic Locking Paket

Das owa_opt Lock Paket wird für die Implementation eines locking Models in PL/SQL Webanwendungen eingesetzt. Bei Datenbankoperationen wird das Locking dazu benutzt, um mehreren Datenbankanwendern daran zu hindern, gleichzeitig Veränderungen an den selben Daten vorzunehmen. Wenn z.B. zwei Benutzer mit dem Select-Befehl dieselben Daten der Datenbank aufrufen, um sie dann zu verändern. Wenn nun einer von ihnen die Daten mit einem Update-Befehl verändert, sind bei dem anderen Benutzer die vorher mit dem Selectbefehl aufgerufenen, also noch nicht veränderten Daten, noch gültig. Verändert dieser die Daten ebenfalls werden die Veränderungen des ersten Benutzers wiederum überschrieben, ohne das er das u.U. erfährt. Oracle hat deswegen eine

automatische Sperrfunktion, die dann in Kraft tritt, wenn ein Datensatz selektiert wird. In diesem Fall können andere Benutzer, die also denselben Datensatz danach selektieren ihn zwar ansehen, nicht aber verändern. Diese „for update“ Funktion verwaltet Oracle automatisch, außer bei den LOB Datentypen (Seite 42), wo ja bei der Initialisierung eines internen LOB die Selectanweisung mit dem ....for update Befehl abgeschlossen werden muß. Das automatische Sperren von Datensätzen (Locking) ist aber nur dann möglich, wenn die Benutzer während ihrer Arbeit an der Datenbank auch mit dieser Verbunden ist (sonst kann das System ja nicht wissen, welcher Benutzer gerade welche Datensätze selektiert hat).

Eine solche persistente Verbindung zwischen den Benutzern und der Datenbank ist aber bei Webanwendungen nicht möglich, da die Verbindung zwischen Datenbank und Benutzer nach jeder Webanfrage wieder unterbrochen wird. Das opt Lock Paket des WAS weißt ein update desselben Datensatzes ab, solange ein anderer Benutzer gerade diese Daten verändert. Ist dieses erste Update abgeschlossen werden die aktualisierten Daten dem zweiten Benutzer angezeigt. Das Original des Datensatzes wird hierzu in einer owa_Lock Routine gespeichert und den HTML Forms als hidden Fields mitgeliefert. Das PL/SQL Programm, welches die Datensatzänderungen durchführen soll benutzt eine opt Lock Routine, um die gespeicherten Informationen mit dem aktuellen Datensatzinformationen zu vergleichen. Dieses geschieht durch einen Checksummenvergleich.

3.3. Das owa Cookie Paket

Das owa Cookie Paket unterstützt die Verwendung von sog. HTML Cookies. Dies sind Informationen, die mit den Antwortdaten des Webservers zum Browser übertragen werden. Im Grunde besteht solch ein Cookie aus sechs Teilen, die ersten drei sind: der Name des Cookies, ein alphanumerischer Wert, welchen das Cookie annehmen kann - man kann sich das ganze wie ein name/value Paar vorstellen und schließlich das Verfallsdatum des Cookies. Diese Cookie geben dem Anwendungsentwickler die Möglichkeit, Benutzer zu identifizieren bzw. frühere Tätigkeiten des Benutzers in der Anwendung nachzuvollziehen. Wenn ein Benutzer beispielsweise zum ersten mal eine Anwendung aufruft, er also noch kein Cookie dieser Anwendung auf seinem Rechner gespeichert hat, z.B. eine virtuelle Kaufhausanwendung, wird dies von der Anwendung erkannt.

Nun könnte die Anwendung dem neuen Benutzer z.B. ein Anmeldeformular zuschicken, wo er seine Daten einträgt und der Anwendung damit bekannt gibt. Die Anwendung kann nun die Daten des neuen Benutzers verarbeiten und beispielsweise in eine Datenbank ablegen. Sie kann nun eine Kundennummer für den Benutzer generieren und ihm dann ein Cookie mit dem Wert der Kundennummer zuschicken. Wenn der Benutzer zu einem späteren Zeitpunkt diese Anwendung wieder Aufruft, erkennt das Programm das Cookie (sofern es noch nicht durch sein Verfallsdatum gelöscht wurde) und schickt ihm nun kein Registrierungsformular mehr zu. Wird beim setzen des Cookies kein Verfallsdatum angegeben, so verfällt es in dem Augenblick, da der Benutzer die Anwendung verläßt. Die weiteren drei Bestandteile, dienen der Sicherheit für das Arbeiten mit Cookies, dies sind: der Sicherheitsstatus, sowie der Pfad und die Internetdomäne, von der aus das Cookie gesetzt wurde. Im Sicherheitsstatus wird

beschrieben, von wo und wann das Cookie gesetzt wurde und eine evtl. Verschlüsselung, wie beispielsweise SSL. Da Cookies als lokale Dateien auf den Benutzerrechner geschrieben werden sollte man bei der Arbeit mit Cookies darauf achten, keine Sicherheitsrelevanten Daten als Cookiewert, wie z.B. Paßwörter, zu setzen. Da verschiedene Cookies von vielen Anwendungen benutzt werden können,, kann der Browser durch den Pfad- und Domänennamen des Cookies die Anwendung, zu dem das Cookie gehört identifizieren, so das nur das zugehörige Cookie an die entsprechende Anwendung zurückgegeben wird. Das owa Cookie Paket verwaltet Cookies mit folgendem Datentyp:


```
TYPE cookie IS RECORD
        {       name      VARCHAR2(4096),
                vals      VC.ARR,
                num-vals  INTEGER
        };
```

Außerdem enthält es Funktionen, um ein Cookie zu setzen, zu identifizieren, um verfügbare Cookies auszuführen und zu Verfallsdaten des Cookies. Diese Prozeduren werden im Antwortkopf des Servers übermittelt und müssen so vor jeder htp oder htf-Prozedur verarbeitet werden. So ist es für den Entwickler wichtig zumindest den Mime Typ im Antwortkopf selbst zu schreiben. Dieses kann man mit einer Prozedur aus dem owa util Paket machen:

owa_util.mime_header(‘text/html’,FALSE);

Mit dem folgenden kurzen Programmbeispiel werden einige der owa Cookie Prozeduren verwendet. Hier wird pro Programmaufruf ein Cookie mit Namen name001 und Wert value001 gesetzt. Dann werden die vorhandenen Cookies ausgegeben. Hierbei ist zu beachten, daß beim ersten Aufruf der Prozedur noch keine Cookies angegeben werden, da sie in derselben Prozedur gesetzt wurden. Erst beim zweiten Aufruf der Prozedur wird das zuerst gesetzte Cookie angezeigt. Die gesetzten Cookies verfallen am jeweiligen Systemdatum + einen Tag. Genauso, wie das setzen von Cookies kann man sie mit der Prozedur owa_cookie.remove(name,value) sofort löschen. Weiterhin gibt es noch eine Prozedur für das Auslesen eines bestimmten Cookies: owa_cookie.get(name).


```
--Cookietest
CREATE OR REPLACE PROCEDURE cookietest
IS
cname owa_cookie.vc_arr; --Variablen für das Auslesen der Cookies
cvalue owa_cookie.vc_arr;
cnumber INTEGER;
x INTEGER; --Zähler
BEGIN
-- Der MIME Typ der Seite muss manuell gesetzt werden
owa_util.mime_header('text/html',FALSE);
--Setzen eines Cookies mit namen name001 und Wert 001, welches einen
--Tag nach dem Systemdatum verfallen soll
owa_cookie.send('name001','value001',SYSDATE+1);
--Schließen des Kopfes
owa_util.http_header_close;
--Auslesen aller gesendeten Cookies
owa_cookie.get_all(cname,cvalue,cnumber);
htp.htmlOpen;
htp.headOpen;
htp.title('Cookies senden und Ausgeben');
htp.headClose;
htp.bodyOpen;
--Ausgabe der Anzahl der gesendeten Cookies
```

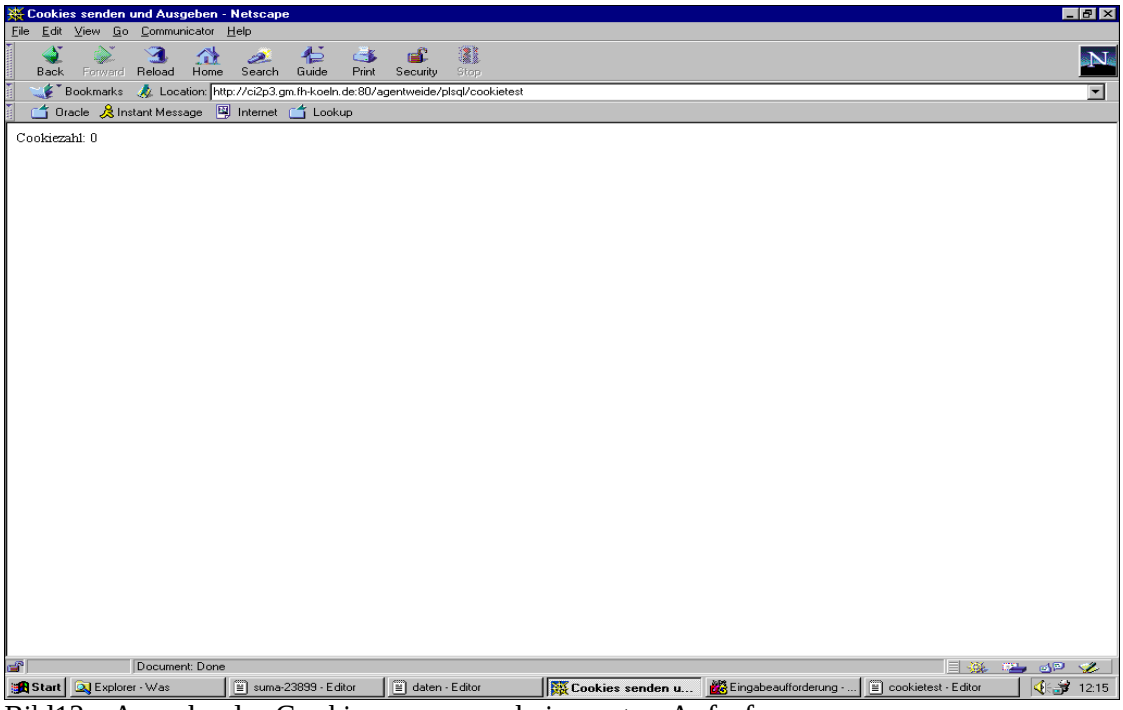


Bild13a: Ausgabe des Cookieprogramms beim ersten Aufruf

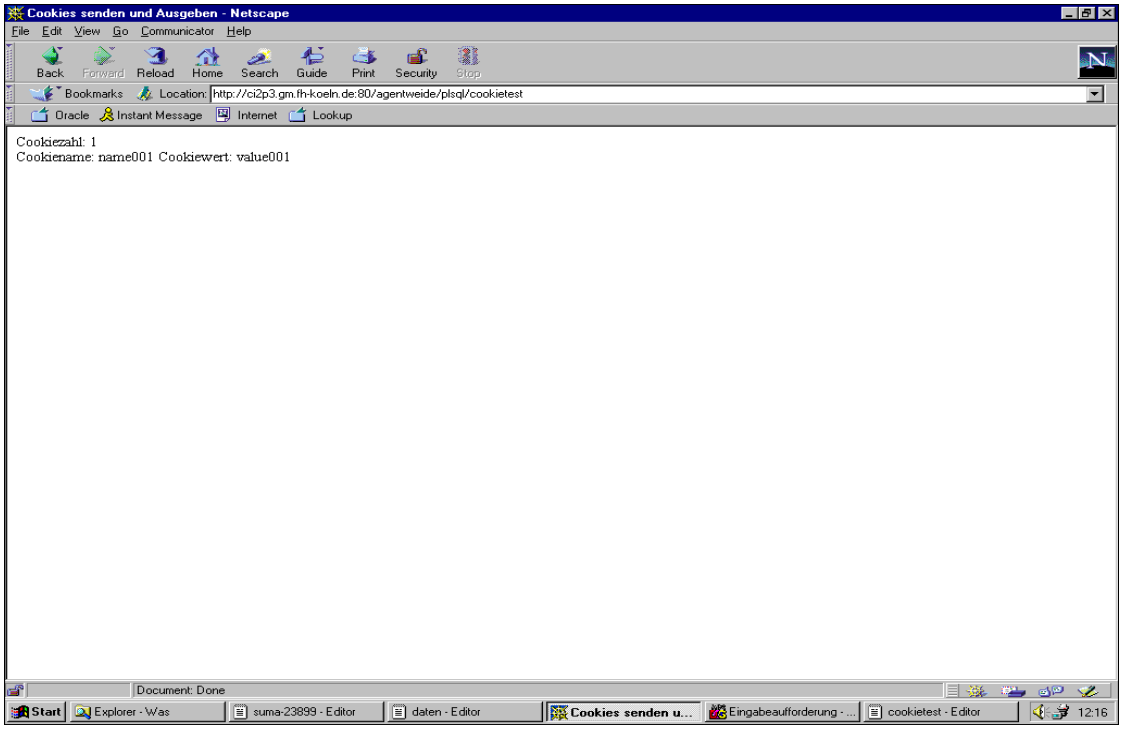


Bild13b: Ausgabe des Cookieprogramms beim zweiten Aufruf

4. Entwicklung einer Beispielanwendung mit dem Web Application Server

4.1. Arten von Suchmaschinen

Die Informationen und Daten im WWW sind so vielfältig und in so großer Menge vorhanden, daß es für jemanden, der eine bestimmte Information sucht schwierig und langwierig ist seine gewünschte Information zu erhalten. Für das gezielte Auffinden von Daten im „Datenwust“ des WWW helfen die Suchmaschinen. Für den Benutzer stellen sich die Suchmaschinen so dar, daß er einen oder mehrere Suchbegriffe eingeben kann und die Suchmaschine alle Webseiten (als URL) ausgibt, die diese(s) Stichwort(e) enthalten. Die Suchmaschine kann natürlich nur die Webseiten berücksichtigen, die sie kennt.

Heute gibt es leider noch keine Suchmaschine, die wirklich alle öffentlichen Webseiten kennt und so alle Webseiten durchsuchen kann. Trotzdem haben die heutigen Suchmaschinen schon so viele Daten gespeichert, daß sie die Suche nach den gewünschten Informationen im WWW erleichtern. Nach der Größe der gespeicherten Informationen (Webseiten) der Suchmaschine und dem Verfahren an diese Webseiten zu kommen, kann man heute drei Arten von Suchmaschinen unterscheiden. Zum einen gibt es die lokalen Suchmaschinen, die nur das Auffinden von Seiten eines (lokalen) Servers bieten z.B. das Suchen nach Webseiten einer Hochschule unterstützen. Solch eine lokale Suchmaschine soll später in diesem Abschnitt als Beispielanwendung mit den Web Application Server vorgestellt werden. Neben den lokalen Suchmaschinen gibt es Suchdienste, die Daten von mehreren Servern gespeichert haben. Dieses sind meist kommerzielle Suchmaschinen (meist durch Werbung oder den Eintrag von Webseiten finanziert).

Die Daten dieser Suchmaschinen können einmal redaktionell aufgenommen sein, d.h. eine menschliche Redaktion stellt die Daten zusammen, gewichtet und bewertet sie und stellt so die Daten für die Suchmaschine zusammen. Daneben gibt es sog. Crawler die automatisch das Netz nach Informationen „absurfen“ und so an die Informationen für die Suchmaschine kommen. Die Suche selbst erfolgt allerdings immer offline d.h. die Daten müssen an irgend einer Stelle der Suchmaschine lokal gespeichert (als Index oder in einer Datenbank) sein. Eine online Suche oft über mehrere Millionen Webseiten würde viel zu lange Dauern. Der Crawler aktualisiert die Daten der Suchmaschinen dagegen von Zeit zu Zeit, so das der Suchdienst immer auf dem aktuellen Stand gehalten wird. Als dritte Kategorie von Suchdiensten kann man die sog. Metasuchmaschinen ansehen.

Dieses sind eigentlich keine echten Suchmaschinen. Sie suchen nicht selbst nach Daten, sondern senden den eingegebenen Suchbegriff gleichzeitig an mehrere ausgewählte Suchmaschinen. Diese suchen dann parallel nach den gewünschten Informationen und senden die Ergebnisse der Metasuchmaschine zu. Diese filtert dann die Daten (sucht doppelte Einträge heraus, gewichtet die Ergebnisse u.ä.) und sendet die so aufbereiteten Daten an den Benutzer. Der Vorteil von Methasuchmaschine ist, daß mehrere Suchmaschinen parallel nach den gewünschten Informationen suchen und so ein wesentlich größerer Datenbestand durchsucht werden kann.

4.2. Beschreibung der Beispielanwendung

Bevor ich mit der Erklärung des Aufbaus des Beispielprogramms beginne, eine kurze Bemerkung zu den verwendeten Betriebssystemen. Der von mir für diese Arbeit installierte Web Application Server lief auf einem Windows NT 4.0 Rechner. Deshalb beziehen sich die Beschreibungen für die Installation und die Administration auf eine Web Applications Server Installation unter NT. Das Beispielprogramm (Suchmaschine) sollte aber auf einem Unix Rechner laufen, da die html Dateien ebenfalls hierauf liegen. Leider stand mir der Unix Rechner nicht für die Programmtests zur Verfügung, so daß ich das Hauptprogramm (suma) und die Funktion konvert auf einem NT Rechner geschrieben und getestet habe. Damit sie in einer Unix Umgebung ebenfalls laufen, habe ich die

Dateipfade umgeschrieben (anstatt C:\Users\Weide\dateien dann /home/iw070/WAS/dateien) und alle Tests auf das \ Zeichen in Tests auf das / Zeichen verändert. Das Unix Script zum sammeln aller html Dateien auf dem Server konnte ich allerdings auf demselben testen. Die relevanten html Dateien befinden sich auf dem Server in zwei Pfaden. Das eine sind html Dateien, die für die Homepage der FH selbst angelegt wurden, sie befinden sich unter dem Pfad: /usr/local/www/htdocs ...
Das zweite sind die Homepages die die Studenten und Mitarbeiter der FH selbst unter ihrem Account anlegen. Diese befinden sich unter den Pfaden:
/home/BENUTZERNAME/public_html/...
wobei BENUTZERNAME für den jeweiligen Benutzeraccount steht, also z.B. iw070. Um diese Dateien zu finden habe ich ein kurzes Unix Script geschrieben, welches diese Pfade untersucht und alle htm und html Dateien mit ihrem vollen Pfad in die Datei daten.txt speichert. Das Script ist im Anhang dieser Arbeit aufgeführt.

4. Entwicklung einer Beispielanwendung mit dem Web Application Server

Hierbei sei noch erwähnt, daß nur die Dateien gefunden werden, die auch öffentlich sind, also nicht lesegeschützt sind. Die Datei daten.txt wird nun von den PL/SQL Programmen verwendet, um die darin enthaltenden Dateien nach einem Suchwort zu durchsuchen. Dieses Script habe ich mit der Hand gestartet, es wäre aber auch vorstellbar Unix anzuweisen, es von Zeit zu Zeit (z.B. alle 24 Stunden) auszuführen, um die Daten zu aktualisieren. Die Datei „daten.txt“ befindet sich in meinem Homeverzeichnis /iw070. Einen kleinen Ausschnitt der Datei ist in der Abbildung 16 (Anhang A) dargestellt. Es handelt sich hierbei um einen kleinen Ausschnitt der Datei, welche ca. 2000 Zeilen (Dateipfade) enthält. Die nun folgende Beschreibung des Hauptprogramms (suma) bezieht sich auf die getestete NT Version der Prozedur (suma). Die Änderungen für die im Anhang

aufgeführte Unix Version beschreibe ich am Ende dieses Kapitels. Das Hauptprogramm (suma) erzeugt zunächst eine Eingabeseite (Form) auf die der Benutzer sein Suchwort eingeben kann. Diese PL/SQL Prozedur ruft sich nach einem Submit mit der Eingabe rekursiv wieder auf. Wurde ein Suchwort eingegeben ließt das Programm (suma) von der Datei daten.txt alle verfügbaren html Dateien (mit Pfad) ein. Zusätzlich trennt es den Dateipfad und den Dateinamen, um die Bfiles nutzen zu können (Bfilezeiger brauchen zwei Eingaben, nämlich eben den Pfad als Verzeichnisalias und den Dateinamen vgl. S40). Der Pfad wird hierbei für das Ändern des physikalischen Dateipfads in der Systemtabelle SYS.DIR$ verwendet, um den Verweis des Aliaspfads zu verändern. Nun werden alle Dateien, die in der Datei daten.txt aufgelistet sind, nach dem Suchwort durchsucht. Wird das Suchwort in einer Datei gefunden, gibt suma den Dateipfad und den Dateinamen als Link aus (Siehe Abbild 15b). Der Unterschied zwischen der NT Testversion und der Unix Version besteht hauptsächlich in der Ausgabe der Suchergebnisse.

4. Entwicklung einer Beispielanwendung mit dem Web Application Server

Die Unix Version soll nicht nur die Dateipfade und Dateinamen ausgeben, sondern die entsprechende URL zu der gefundenen Datei als Link ausgeben. Hierzu habe ich eine Funktion „konvert“ geschrieben, welche die Dateipfade in die entsprechende URL umwandelt. Wird also in einer Datei das Suchwort gefunden, ruft das Hauptprogramm (suma) die Funktion konvert auf, um auf der Grundlage des Dateinamens und Dateipfades die gültige URL dieser html Datei zu erstellen. Hierzu wird die Funktion „konvert“, mit dem kompletten Dateipfad des html Files als Argument, aufgerufen. Der Dateipfad wird dann, je nachdem ob es sich um eine Homepage eines Benutzers oder um die Homepageseiten der FH handelt, in die entsprechende URL umgewandelt. Alle Homepages der Benutzer haben im Dateipfad den Eintrag /home/, da sich die html Dateien im

jeweiligen Homeverzeichnis befinden. Dieser Eintrag wird durch „http://www.gm.fh-koeln.de/~“ ersetzt. Weiterhin sind diese Dateien unter dem Pfad /public_html/ abgelegt, da der Schuldeamon nur Homepages der Benutzer unter diesem Pfad listet. Dieser Eintrag wird aus dem Dateipfad entfernt. Als letztes wird der Dateianhang entfernt. Das Ergebnis ist die entsprechende URL. Nehmen wir an, daß ein Benutzer eine html Datei unter dem Pfad: „/home/iw070/public_html/index/index.htm“ gespeichert hat. Die Funktion „konvert“ erkennt die Schlüsselangabe /home/ und wandelt den Pfad, wie oben beschrieben um. Als erstes wird der Eintrag „/home/“ ersetzt. Der veränderte Pfad lautet jetzt:

„http://www.gm.fh-koeln.de/~iw070/public_html/index/index.htm“

Im zweiten Schritt wird die Angabe „/public_html“ entfernt. Zum Schluß wird der Dateianhänge entfernt, so daß die fertige URL so aussieht:

„http://www.gm.fh-koeln.de/~iw070/index/index“.

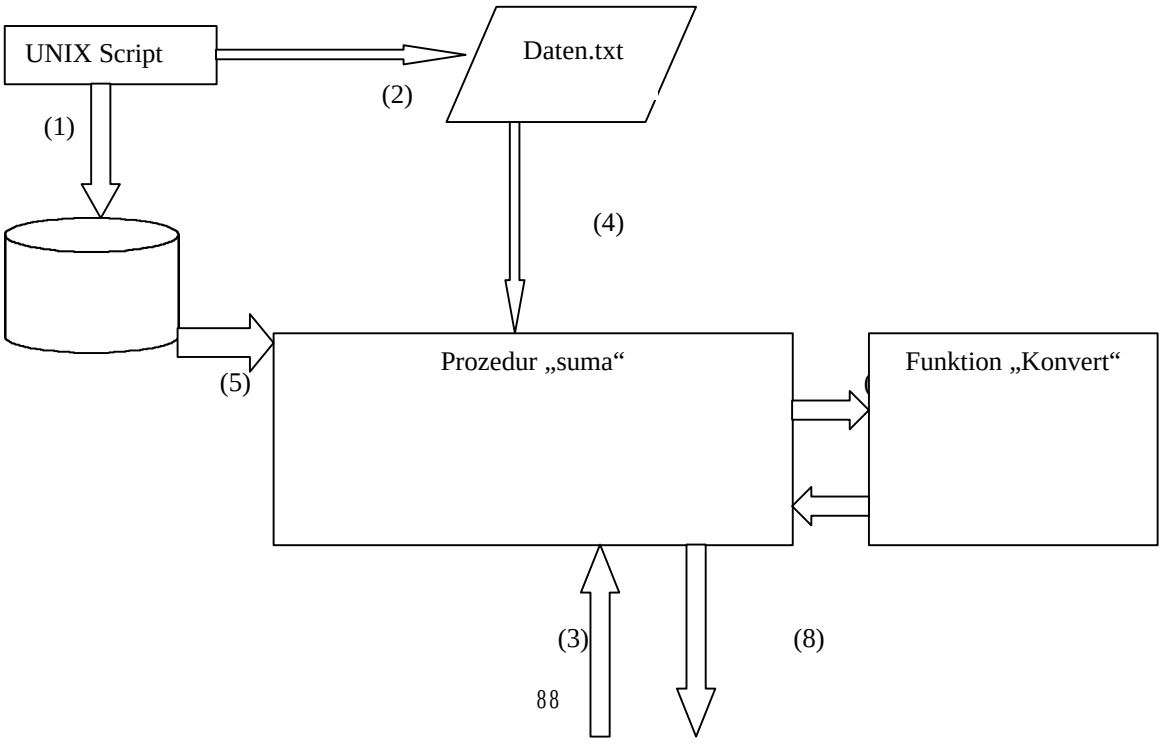
Mit diesem Verfahren lassen sich alle Homepage html-Dateipfade der Benutzer in die dazugehörige URL umwandeln.

4. Entwicklung einer Beispielanwendung mit dem Web Application Server

Das Verfahren für die Umwandlung von FH-Homepage-Dateien sieht etwas anders aus, da diese Dateien unter dem Pfad „/usr/local/www/htdocs/“ untergebracht sind. Nehmen wir gleich das Beispiel der html Datei „/usr/local/www/htdocs/welcome.htm“. Bei allen FH-Homepage-Dateien, die unter diesem Pfad liegen, wird der Eintrag

„/usr/local/www/htdocs/“ durch „http://www.gm.fh.koeln.de/“ ersetzt. Dann, im zweiten Schritt, wird der Dateianhang entfernt. Aus dem Dateipfad wird so die URL „http://www.gm.fh-koeln.de/welcome“. Die weiteren Unterschiede zwischen der NT Version und der Unix Version bestehen dann nur noch in der Verarbeitung von Pfadangaben. Die Pfadangaben in der Directorydefinition (vgl.

Anhang A Punkt2 „CREATE DIRECTORY...) und im Quellcode des Hauptprogramms (suma - Anhang A Definition von orgverz) mit dem Pfadeintrag „/home/iw070/WAS/dateien/“ müssen bei der Implementation der Programme auf den Pfad gesetzt werden, in den das Unix Script die Datei „daten.txt“ hineinschreibt. Für die NT Version, in der mir das Unix Script ja nicht zu Verfügung stand, habe ich per Hand eine entsprechende Datei geschrieben, die natürlich NT bzw. DOS Pfadangaben enthielt. Ansonsten war bei der Verarbeitung von Pfadangaben in der NT Version natürlich das „\“ Zeichen für die Trennung der Verzeichnisse verantwortlich. In der Unix Version wurden die Test auf dieses Zeichen in Tests auf das „/“ Zeichen umgeschrieben. Außer dem Zugriff auf Betriebssystemdateien mit Hilfe von Bfiles werden alle Operationen der Programme innerhalb des Oracle Datenbanksystems ausgeführt. Meist handelt es sich dabei um simple Zeichenkettenoperationen, so daß die im Anhang A aufgeführte Unix Version ohne Probleme auch innerhalb einer Unix Umgebung laufen sollte. Das Vorgehen bei der Implementation der Programme habe ich im Abschnitt 4.3. beschrieben. Einen Überblick über die Programmteile und –abläufe sind im folgenden Abbild noch einmal graphisch aufgezeigt.



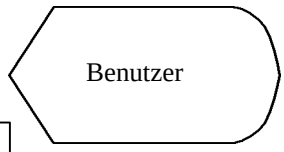


Bild 14 : Ablauf der Beispielanwendung

(1) Das Unix Script durch sucht die Pfade /home/accountname/public_html und /usr/local/www/htdocs nach html Dateien
(2) und speichert die Pfade der gefundenen Dateien in die Datei „daten.txt“.
(3) Der Benutzer gibt sein Suchwort ein.
(4) „suma“ geht die Dateien, deren Verzeichnispfade in der Datei „daten.txt“ gespeichert sind zeilenweise durch
(5) öffnet die Dateien nacheinander und vergleicht den Inhalt der Dateien mit dem Suchwort.
(6) Wird das Suchwort gefunden wird der Pfad der entsprechenden Datei an die Funktion „konvert“ gegebe, die aus dem Pfadnamen eine URL macht
(7) Die URL wird an die Hauptprozedur zurückgegeben.
(8) Dem Benutzer werden die Ergebnisse als Link ausgegeben. Die Vorgänge (5) bis (7) wiederholen sich bis EOF der Datei „daten.txt“ erreicht ist.

4.3. Implementation

Für die Implementation der Programme sollte auf einem UNIX Rechner eine Oracle 8 Datenbank installiert sein und ein Benutzer für die Verwendung der Prozeduren eingerichtet sein. Der Web Application Server sollte installiert sein und wie im Kapitel 2.3 beschrieben für den Benutzer der DAD und die PL/SQL Pakete installiert sein. Zusätzlich braucht der Benutzer update Rechte für die Systemtabelle SYS.DIR$. Als erstes sollte wie im Kapitel 1.3.3.6.2. beschrieben die Tabellen für die Initialisierung von BLOBs angelegt werden. Danach kann man die CREATE DIRECTORY, SUMA und KONVERT Prozeduren auf den Rechner kopieren. Alle drei Programme lassen sich mit dem @ Befehl in die Datenbank ablegen. Es ist darauf zu achten, daß die Pfade in dem CREATE

DIRECTORY Befehl und in der Zuweisung der Variable „orgverz“ des SUMA Programms mit dem Pfad der durch das Unix Script erstellten Datei „daten.txt“ übereinstimmen. Dieses Unix Script kann nun kopiert und gestartet werden. Das Hauptprogramm kann dann durch die URL wie im Kapitel 3.0 beschrieben aufgerufen werden bzw. es kann natürlich auch von einer anderen Seite mit einem Link auf diese URL gestartet werden.

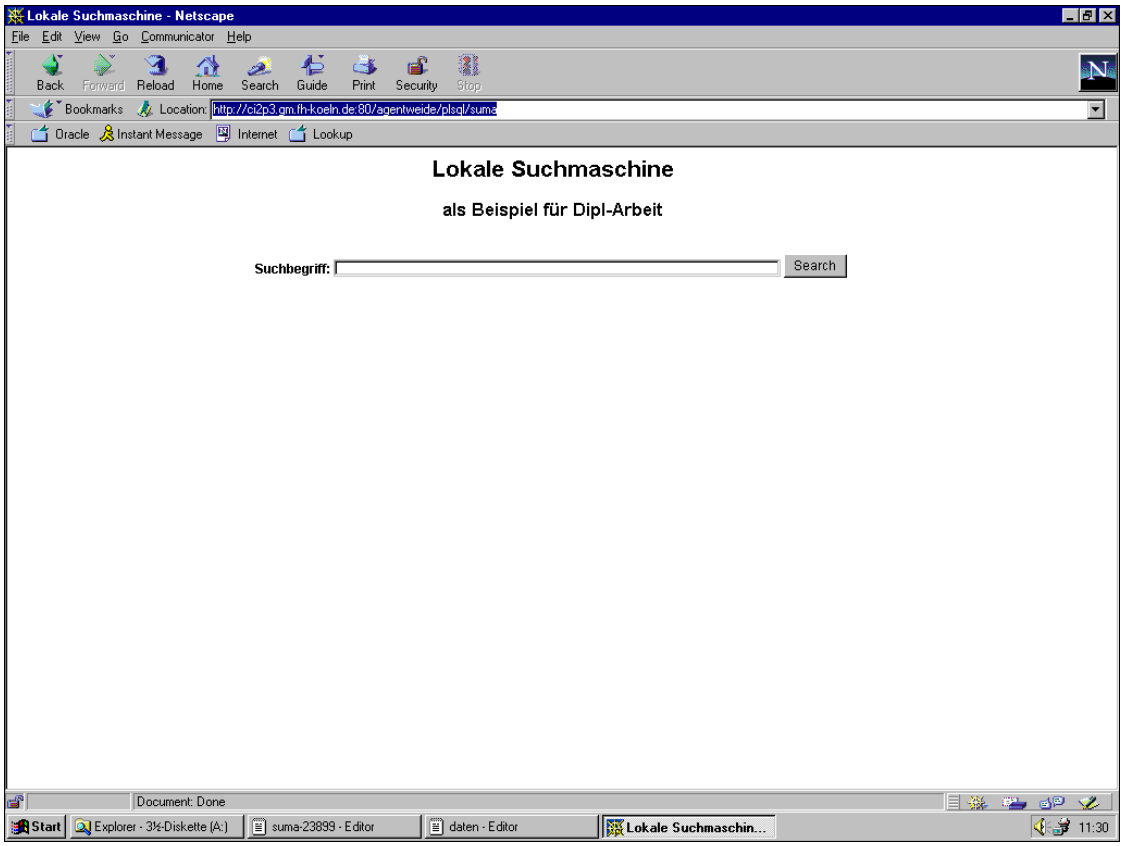


Bild15a: Ausgabe des suma Programms

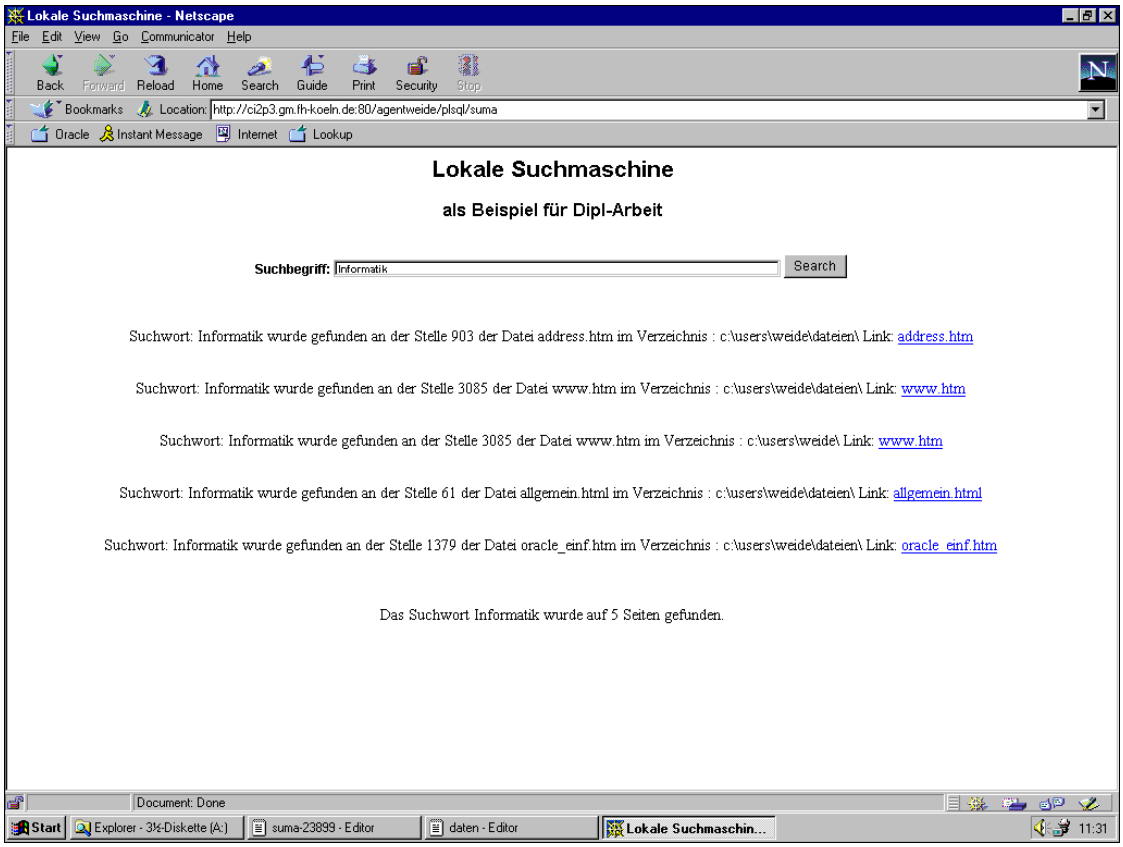


Bild15b: Ausgabe von suma nach der Eingabe eines Suchwortes (NT Testversion)

Anhang A : Dokumentierter Quellcode des Beispielprogramms

1. Das Unix Script für das Auffinden der html Dateien und das Abspeichern der Dateipfade in die Datei „daten.txt“

```
rm /home/iw070/WAS/dateien/daten.txt
find /usr/local/www/htdocs –name ‘*.html‘ –print > /home/iw070/WAS/dateien/daten.txt
find /usr/local/www/htdocs –name ‘*.htm‘ –print >> /home/iw070/WAS/dateien/daten.txt
find /home/*/public_html –name ‘*.html‘ –print >> /home/iw070/WAS/dateien/daten.txt
find /home/*/public_html –name ‘*.htm‘ –print >> /home/iw070/WAS/dateien/daten.txt
```

Die erste Zeile löscht die vorhandene Datei. In der zweiten und dritten Zeile werden die Pfade unter /usr/local/www/htdocs/... nach Dateien mit der Endung *.htm oder *.html durchsucht, und die gefundenen Daten in die Datei „daten.txt“ mit ihrem vollständigen Pfad gespeichert. Alle hier gefundenen Dateien gehören zur Homepage der FH. Die letzten beiden Zeilen durchsuchen die Verzeichnisse im Oberverzeichnis /home/ unter dem Unterverzeichnis /public_html alle .htm bzw .html Dateien und speichern die Dateipfade ebenfalls in der Datei „daten.txt“ ab. Bei diesen html Dateien handelt es sich um die Homepages der Benutzer.

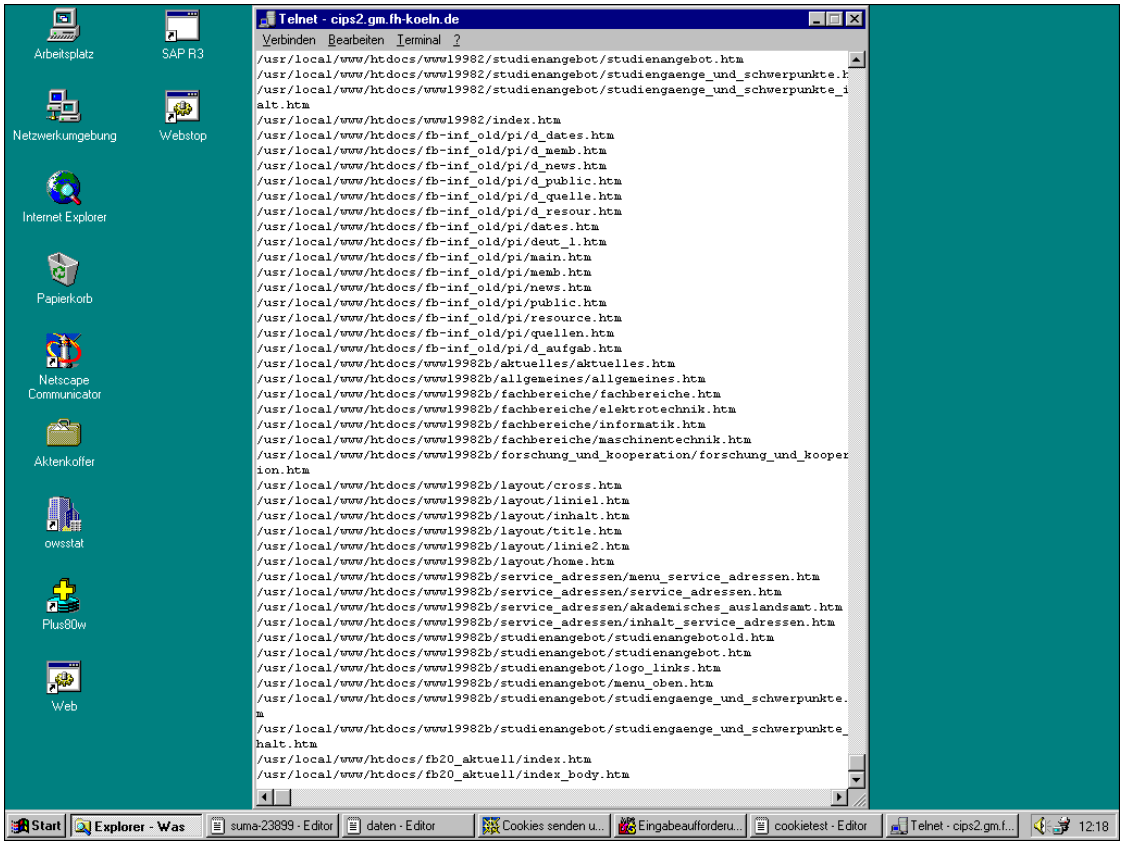


Bild16: Ausschnitt aus der Datei daten.txt, die vom Unix Script erzeugt wurde

2. Create Directory für das Erstellen eines Verzeichnisalias, um Dateien mit der Hilfe von Bfiles auslesen zu können.:

```
--Verzeichnisallias für einen Externen Datentype (BFILE) erstellen. Die folgende Zeile
--erstellt ein Verzeichnisallias unter dem die BFILE Dateien abgelegt sind
CREATE OR REPLACE DIRECTORY "htmfiles" AS '/HOME/IW070/WAS/dateien/';
```

3. Für die Vereinfachung der Initialisierung der Clobs wird folgende Tabelle angelegt: (Vgl. S.43)

```
CREATE TABLE temp_blob ( temp_blob      BLOB,
                         temp_clob      CLOB,
                         temp_nclob     NCLOB);
```

Und mit folgenden Werten gefüllt: (Vgl. S. 43,44)

```
INSERT INTO temp_blob VALUES(
empty_blob(), empty_clob(), empty_nclob());
```

Das Hauptprogramm (suma): (Seite 1)

```
CREATE OR REPLACE PROCEDURE suma (text in VARCHAR2:=NULL)
IS
--Orginalpfad des Verzeichnisalias „htmfiles“
orgverz    VARCHAR2(1000):='/HOME/IW070/WAS/dateien/';
--Änderungen des Pfads werden in der „neuverz“ Variable gespeichert
neuverz    VARCHAR2(1000);
--Pfadliste der html Dateien befindet sich in der Datei daten.txt
datei    BFILE:=bfilename('htmfiles','daten.txt');
--Bfile Zeiger für die zu Untersuchenden html Dateien
temp    BFILE;
--Zähler
counter INTEGER;
-- „bf_name“ speichert Dateiname
bf_name VARCHAR2(50);
-- „bf_dir speichert Dateipfad
bf_dir  VARCHAR2(50);
-- „neu“ speichert das Suchwort in html Format (Sonderzeichen)
neu      VARCHAR2(50);
-- „len“ speichert die Länge der einzul. Dateiliste
len      INTEGER;
-- „leng“ speichert die Länge der einzulesenden html Datei
leng     INTEGER;
--Speichert die Gesammtlänge des auszulesenden Pfades
gesamt          INTEGER;
--Speichert letztes / Zeichen für Trennung des Pfades und Dateinamen
pos      INTEGER;
--Speichert Korrekturlänge des fortlaufenden Zählers
kor      INTEGER:=0;
```

Das Hauptprogramm (suma): (Seite 2)

```
--Speichert Zeichenanzahl bis zum letzten / Zeichen des Pfades
anz     INTEGER;
--Speichert Offset (Anfangsbuchstabe) des gefundenen Suchwortes
fndset  INTEGER:=0;
--Speichert Anzahl der gefundenen Seiten mit dem Suchwort
z       INTEGER;
--Speichert den Inhalt der Datei daten.txt (Pfadliste)
daten   CLOB;
--Speichert den Inhalt der aktuellen html Datei
cl2     CLOB;
--Speichert die Zeichen beim Auslesen des Clobs
p       CHAR(1);
--Parameter für die dbms_lob.read Funktion
offset  INTEGER:=1;
--Speichert den kopletten Dateipfad
path    VARCHAR2(100);
--Speichert nur den Pfad (Ohne Dateinamen)
verz    VARCHAR2(100);
--Speichert nur den Dateinamen
datname VARCHAR2(100);
--cr speichert den Inhalt des <CR>
cr      VARCHAR2(3):='
';
--„link“ speichert die URL
link    VARCHAR2(1000);
BEGIN
--„neuverz“ erhält den gleichen Pfad wie „orgverz“
neuverz:=orgverz;
--Initialisierung der beiden zu verwendenden CLOBS mithilfe einer Tabelle
SELECT temp_clob INTO daten FROM temp_blob FOR UPDATE;
SELECT temp_clob INTO cl2 FROM temp_blob FOR UPDATE;
--Öffnen der Ausgabe (html)
htp.htmlOpen;
```


Das Hauptprogramm (suma): (Seite 3)

```
--Erstellung des Seitentitels
htp.headOpen;
htp.title('Lokale Suchmaschine');
htp.headClose;
htp.bodyOpen;
--Textformat und Überschriften der Seite
htp.p('<FONT FACE="ARIAL" SIZE=2 COLOR="#000000">');
htp.p('<h2><center><a NAME="1">Lokale Suchmaschine</a></center></h2>');
htp.p('<b>');
htp.p('<center><h3>als Beispiel für Dipl-Arbeit</h3></center>');
htp.p('<BR>');
--Texteingabeform für die Eingabe eines Suchbefriffes
--Das Form ruft dieses Programm rekursiv mit Suchbegriff als Parameter wieder
--auf
htp.formOpen('suma');
htp.p('<center>');
htp.p('Suchbegriff:');
--Suchbegriff wird in der "Name-Variable" text (VARCHAR2-Parameter dieser
-- Prozedur) gespeichert
htp.formText('text',50,50,text);
htp.p('</b>');
--Erstellt den Submitbutton
htp.formSubmit(NULL,'Search');
htp.p('</center>');
--Beendet das Texteingabeformular
htp.formClose;
htp.p('</FONT>');
htp.p('<br>');
--Umwandeln des Suchstrings in die html Form für deutsche
--Sonderzeichen
neu:=REPLACE(text,'ä','&auml;');
neu:=REPLACE(neu,'ö','&ouml;');
neu:=REPLACE(neu,'ü','&uuml;');
neu:=REPLACE(neu,'Ä','&Auml;');
neu:=REPLACE(neu,'Ö','&Ouml;');
neu:=REPLACE(neu,'Ü','&Uuml;');
neu:=REPLACE(neu,'ß','&szlig;');
```


Das Hauptprogramm (suma): (Seite 4)

```
--Wurde diese Prozedur mit einem Eingabewort aufgerufen wird hier fortgesetzt
--sonst wird die Prozedur beendet
IF text IS NOT NULL THEN
  z:=0;
--Öffnen des BFILES datei, um Pfadliste der zu untersuchenden Dateien einzul.
  dbms_lob.fileopen(datei,dbms_lob.file_readonly);
  --Ermittlung der Länge des BFILES datei
  len:=dbms_lob.getlength(datei);
  --Einlesen des Dateiinhaltes in einen internen CLOB daten
  dbms_lob.LOADFROMFILE(daten,datei,len);
  --schließen der Dateilistendatei "datei"
  dbms_lob.fileclose(datei);
  --Zähler auf 1 setzen
  counter:=1;
  --Korrecturzähler auf 0 setzen
  kor:=0;
  --Beginn der Ausleseschleife
    LOOP
    --Auslesen des CLOBS "daten" Zeichen für Zeichen bis das <CR> erscheint
    dbms_lob.read(daten,offset,counter,p);
    IF p!=cr THEN path:=path||p;
      counter:=counter+1;
      --Einlesen der Zeichen in die Textvariable path solange bis p=<CR> ist bzw
      -- die Zeile ausgelesen ist
    END IF;
    --Speichert die Position des zuletzt eingelesenen / Zeichens. für die Trennung
    --des Pfads und des Dateinamens später.
    IF p='/' THEN pos:=counter-kor;
    END IF;
    --Wenn eine Zeile ausgelesen wurde (das <CR> Zeichen gefunden wurde)
    --gets hier weiter
    IF p=cr THEN
      --der Zähler wird um eins erhöht, da das CR Zeichen aus 2 Stellen besteht
      counter:=counter+1;
      --Ermittlung der Anzahl der Zeichen in der Variable path
      gesamt:=length(path);
      --Ermittlung der Pfadlänge bis zum letzten /
      anz:=gesamt-pos;
      --Ermittlung der Korrekturlänge
      kor:=counter-1;
```

```
--Speichern des Gesamtpfads in das Verzeichnis(verz) und den
--Dateinamen(datname)
datname:=SUBSTR(path,pos,anz);
verz:=SUBSTR(path,1,gesamt-anz-1);
--Ändern des Orginalpfades in der Systemtabelle sys.DIR$ , um zugriff zu
-- haben
UPDATE SYS.DIR$ SET OS_PATH = verz WHERE OS_PATH =
neuverz;
--Änderung speichern
neuverz:=verz;
--Speichern des BFILES mit dem Extraierten Dateinamen
temp:=bfilename('htmfiles',datname);
--Öffnen des BFILES zum lesen
dbms_lob.fileopen(temp,dbms_lob.file_readonly);
--Ermittlung der Länge des BFILE Inhalts
leng:=dbms_lob.getlength(temp);
--Initialisierung eines CLOBS
SELECT temp_clob INTO cl2 FROM temp_blob FOR UPDATE;
--Laden des BFILES in den CLOB cl2
dbms_lob.LOADFROMFILE(cl2,temp,leng);
--Kürzen der CLOB länge auf die länge des Inhaltes
dbms_lob.TRIM(cl2,leng);
--Suchen nach dem Schlüsselwort und Ausgabe des Offsets, an welcher
--Stelle das Wort gefunden wurde
fndset:=dbms_lob.instr(cl2,neu);
--Für den Fall, daß doch jemand deutsche Sonderzeichen benutzt hat
IF fndset=0 THEN fndset:=dbms_lob.instr(cl2,text); END IF;
--Löschen des CLOBS
dbms_lob.ERASE(cl2,leng,1);
--Kürzen des gelöschten CLOBS auf die Länge 0
dbms_lob.TRIM(cl2,0);
--Ist der offset = 0 wurde das Suchwort nicht gefunden ansonsten Ausgabe
--Das Suchwort wurde gefunden an der Stelle fndset bzw Ausgabe des
--Worts, Verzeichnis und Dateiname.
IF fndset!=0 THEN
  z:=z+1;
--Umwandeln des Dateipfades in die URL mit der Funktion „konvert“
  link:=konvert(verz);
  htp.p('<br><center>Suchwort: '||neu||' wurde gefunden');
  htp.p('Die URL lautet:‘);
  --gibt die URL als Link aus
  htp.anchor2(link,link);
  htp.p('</center><br>');
END IF;
```


Das Hauptprogramm (suma): (Seite 6)

```
        --Schließen des temp BFILES
        dbms_lob.fileclose(temp);
        --Zurücksetzen der Pfadangabe
       path:='';
     END IF;
     --Verlassen der Schleife, wenn Ende der Dateiliste erreicht
   EXIT WHEN counter=len+1;
   END LOOP;
   htp.p('<br>');
   htp.p('<line>');
   htp.p('<br><center>Das Suchwort '||neu||' wurde auf '||z||' Seiten
   gefunden.</center>');
   END IF;
htp.bodyClose;
htp.htmlClose;
--Orginalzustand des Verzeichnisalias (SYS-Tabelle) wieder herstellen
UPDATE sys.DIR$ SET OS_PATH = orgverz WHERE OS_PATH = neuverz;
END;
/
show errors
```

Die Funktion „konvert“ für die Umwandlung des Dateipfades in die URL

```
--Funktion konvert für das Umwandeln des UNIX Pfades der gefundenen Datei in
--den entsprechenden virtuellen Pfad (URL) auf dem Schulserver
CREATE OR REPLACE FUNCTION konvert (path IN VARCHAR2)
RETURN VARCHAR2
IS
--Speichert Offset der Dateiendung
point INTEGER := 0;
--Veränderung der Zeichenkette in die Variable neupath speichern
neupath VARCHAR2(1000);
BEGIN
--Fängt der Pfad mit /HOME an, handelt es sich um eine Homepage
--eines Benutzers
IF path LIKE ‘/home/%‘ THEN
--Die Zeichen /home/ werden durch die Zeichen
--http://www.gm.fh-koeln.de/~ ersetzt
```


Die Funktion konvert: (Seite 2)

```
  neupath:= REPLACE(path,'/home/','http://www.gm.fh-koeln.de/~');
  --Die obligatorische Zeichenkette /public_html wird entfernt
  neupath:= REPLACE(neupath,'/public_html','');
  --Die Position des Dateianhangs wird ermittelt (.htm oder .html)
  point:=INSTR(neupath,'.htm');
  --Alle Zeichen nach dem Dateinamen werden entfernt, URL fertig
  neupath:=SUBSTR(neupath,1,point – 1);
ELSE
  --Handelt es sich nicht um die Homepage eines Benutzers, dann ist es
  --eine Seite der Homepage der FH. Diese befinden sich unter dem
  --Pfad /usr/local/www/htdocs/... auf dem Server
  --Dieser Pfadteil wird durch http:/www.gm.fh-koeln.de/ ersetzt
  neupath:=REPLACE(path,'/usr/local/www/htdocs/',
  'http://www.gm.fh-koeln.de/');
  --Ermittlung der Position des Dateianhangs
  point:=INSTR(neupath,'.htm');
  --Alle Zeichen nach dem Dateinamen löschen
  neupath:=SUBSTR(neupath,1,point – 1);
END IF;
--Rückgabe der ermittelten URL
RETURN neupath;
END;
/
show errors
```

Anhang B : Abbildungsverzeichnis

Anhang C : Literaturverzeichnis

Literaturverzeichnis

(1) Internet, Im weltweiten Netz gezielt Informationen sammeln, von Martin Kimmig, dtv Verlag, ISBN: 3 423 50175 6

(2) Oracle 8, Datenbankentwicklung, von David Lockman,SAMS, ISBN:3 8272 2017 3

(3) Oracle, Web Application Server Handbook, Osborne, McGraw-Hill, ISBN: 0 07 882215

(4) Onlinedokumentation des Web Appliaction Servers, sowie der Oracle Datenbank 8

<CR> Zeichen für den Zeilenvorschub.

Account Benutzerkonto mit Zugriffsrechten auf bestimmte Dateien oder Tabellen in einem Betriebssystem oder einer Datenbank.

Administration Systemeinstellungen verwalten.

Aliasnamen Eine weitere Bezeichnung für z.B. ein bestimmtes Directory o. eine Datenbank.

Argument Wert, der an eine Prozedur oder Funktion zur Verarbeitung übergeben wird.

Batchfile Stapelverarbeitungdatei, die darin enthaltenen Anweisung werden beim Aufruf des Batchfiles abgearbeitet.

Browser Programm zur Visualisierung von WWW Inhalten und zur Navigation durch das WWW.

Cartridge Programm zur Unterstützung einer Programmiersprache (z.B. Java,C,PL/SQL), zur Verwendung mit dem WAS.

CGI-Script Common Gateway Interface, Unix Standart für die Client Server Kommunikation.

Checkbox Menü, in dem der Benutzer ein oder mehrere Punkte markieren kann.

Compilieren Umwandlung eines Quellcodes (Text) in ein Maschinenlesbares Format (Binär).

Client Rechner , der die Dienste eines Servers in Anspruch nimmt.

Cookie Informationen, die auf einem Clientrechner

	durch eine Webanwendung übertragen und gespeichert werden.
DAD	Database Access Descriptor enthält Informationen für den Zugriff auf einen Datenbankaccount.
Datenbank	System zum Speichern und Verwalten von Daten.
Datentyp	Ausprägung von Daten z.B. Zeichen, Zahlen oder Binärdaten.
DBA	Der Datenbankadministrator verwaltet eine Datenbank. Er kann z.B. einzelnen Benutzer Zugriffsberechtigungen erteilen oder entziehen
DCL	Data Control Language, Teil von SQL für die Benutzerverwaltung einer DB.
DDL	Data Definition Language, Teil von SQL für die Definition von Datenstrukturen einer DB.

Deamon	Auch Weblistener, Programm zur Verwaltung von Internetresourcen und Bereitstellung dieser im Netz.
dekrementieren	Verkleinerung eines Wertes um einen bestimmten Betrag – Gegenteil von inkrementieren..
Directory	Verzeichnis oder Pfad in einem Dateisystem.
DML	Data Manipulation Language, Teil von SQL für das Abrufen und Verändern von Daten in einer DB.
DOS Fenster (Box)	Fenster unter Windows Betriebssystemen, in dem DOS (Disk Operation System) -befehle oder -programme ausgeführt werden können.
Drop-Down Menü	Aufklappbares Menü in dem der Benutzer einen Menüpunkt auswählen kann.
EOF	End Of File Dateiformatierung, welche das Ende einer Datei anzeigt.
Fremdschlüssel	Verweis auf eine andere Relation, muß gleichzeitig Primärschlüssel der Relation sein, auf die verwiesen wird.
Funktion	Von einem Programm oder Benutzer aufrufbares Programm, welchem Argumente (Werte) übergeben werden können, diese Verarbeitet und einen Wert (Ergebnis) an die aufrufende Instanz zurückliefert.
get	Methode des HTTP , für die Verwaltung von

	Webresourcen.
head	siehe get.
HTML	Hypertext Markup Language : besteht aus Text und Anweisungen (Tags) für die Textformatierungen in WWW Dokumenten.
HTTP	Hypertext Transfer Protokoll, für den Datenaustausch im WWW.
Hyperreference	Referenziert eine andere Webresource mit Hilfe eines Links auf diese.
inkrementieren	Vergrößerung eines Wertes um einen bestimmten Betrag – Gegenteil von dekrementieren.
Internet	Weltweite Vernetzung von Rechnern mit Hilfe des TCP/IP Protokolls. Besteht aus Diensten, wie WWW oder FTP.

IP-Adresse	Eindeutige Adresse mit deren Hilfe das TCP/IP Protokoll Hardware im Netz identifizieren kann.
Library	Standardprozeduren und -funktionen bzw. vorgefertigte Programme für die Anwendungsentwicklung.
Link	Verweis auf eine andere Webresource.
Listener	siehe Deamon.
LOB	Large Object : Datentypen für das Verwalten großer Datenmengen, ab der Oracle 8 Datenbank verfügbar.
Locking	Verfahren, um das Verändern von Daten zu sperren, wenn mehrere Benutzer gleichzeitig denselben Datensatz verändern wollen.
Mime Type	Datentypen, für die Verwendung im http Protokoll.
Netzwerk	Verbund von mehreren Rechnern, die untereinander Daten austauschen können.
NULL	Nicht definierter Wert.
Offset	Position eines Zeichens innerhalb einer Datei oder Clobs.
owsctl [start|stop|status]	Anweisung für das Starten, Stoppen und Anzeigen der WAS Dienste.
Paket	Siehe Library
Parameter	Siehe Argument.
Pfad	

-physikalischer Ordnungskriterium, unter der eine Datei auf einem Speichermedium abgelegt ist.

-virtueller Ordnungskriterium, unter der eine Web-resource vom Weblistener verwaltet wird, bzw. im WWW zu finden ist.

PL/SQL Programmiersprache für Oracle Datenbanken.

Port Anschluß, an den ein Weblistener seine Ein- und Ausgaben bekommt/sendet.

Post Siehe get.

Primärschlüssel Eindeutiger Wert eines Datensatzes, mit dessen Hilfe er identifiziert wird.

Programmiersprache

-prozedural Programmiersprache, die Daten mit Hilfe von Prozeduren und Funktionen verarbeitet, wie z.B. C, Pasacal.

-objektorientiert Programmiersprache, die Daten mit Hilfe von Klassen und Objekten verarbeitet, wie C++.

Prozedur Vom Konzept ähnlich einer Funktion -es werden allerdings keine Daten an die aufrufende Instanz zurückgegeben.

Puffer „Auffangstelle“ für das Zwischenspeichern von Daten.

Quellcode Textdatei mit den Anweisungen für ein Programm, welches später durch das Compilieren in ein computerlesbares Format überführt werden kann.

Radiobox Menüart, ähnlich einer Checkbox, in der Allerdings nur ein Menüpunkt ausgewählt werden muß.

Redundanz Mehrfaches und damit überflüssiges Vorhandensein von denselben Daten und Informationen in einer DB.

Rekursivität Sich selbst aufrufendes Programmteil.

Relation In Beziehung gesetzt. Die Daten in einer relationalen DB werden durch Tabellen in Beziehung zueinander gesetzt.

Router Netzknotenrechner, verteilt Daten in einem Netzwerk.

Schnittstelle Berührungspunkt zwischen zwei Systemen.

Server Anbieter von Diensten in einem Netzwerk.

SQL Structured Query Language oder strukturierte Abfragesprache, für die Verwendung mit Relationalen Datenbanken.

SSL Secure Socked Layer – Layer Protokoll für die

	verlüsselte Sendung von Daten an einen eigenen, von SSL unterstützten Port.
Submit	Versenden von Daten an eine Webanwendung durch den Benutzer.
Suchmaschine	Programm für das Auffinden von Webresourcen, überlichweise durch die Eingabe eines Suchbegriffs.
Tags	Formatierungsanweisungen in einem html Dokument, durch < und > Zeichen begrenzt.
TCP/IP	Transportprotokoll der Internetdienste.
Unix	Betriebssystem.

Webagent	Von einem Cartridge unterstützte WAS Paketinstallation, für einen Datenbankbenutzer.
Windows NT	Ein Betriebssystem der Fa. Microsoft.
WRB	Web Request Brooker., ein Dienst des WAS für das Verteilen von Webanfragen.
WWW	World Wide Web, ein Internetdienst.

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W

Z